

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**DECISION SUPPORT FOR SOFTWARE PROCESS
MANAGEMENT TEAMS:
AN INTELLIGENT SOFTWARE AGENT APPROACH**

by

Lori A. Church

March 2000

Thesis Advisor:
Associate Advisor:

James Bret Michael
John Osmundson

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 1

20000525 059

REPORT DOCUMENTATION PAGE			<i>Form Approved</i> OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 2000		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE DECISION SUPPORT FOR SOFTWARE PROCESS MANAGEMENT TEAMS: AN INTELLIGENT SOFTWARE AGENT APPROACH			5. FUNDING NUMBERS	
6. AUTHOR(S) Church, Lori A.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT Currently, SPAWAR Systems Center is lacking a unified software development environment that allows software developers to effectively manage software development projects across a diversified development environment. This unified environment is needed to provide up-to-date accurate information to the right people at the right time, increase the process knowledge-base, increase productivity, decrease time to market, eliminate redundancy, and ease job stress. This thesis proposes a conceptual model for software process management decision support in the form of an intelligent software agent network. The intelligent software agent network, called MENTOR, provides the knowledge-base that is crucial to the software development team, providing for a repeatable, defined, managed, and optimized development environment. This concept provides SSC software development managers and team members with the ability to work in a unified and collaborative environment, regardless of organizational diversity or location. MENTOR will be utilized as an integral software development team member, providing tutorials and mentoring capabilities for management and process assistance, as well as providing process planning, risk analysis, and strategic planning recommendations for the successful completion of a software development effort, at all team levels. In addition, MENTOR will provide an effective communication environment that will enable the development team to minimize the time consuming workload involved in tracking individual tasking.				
14. SUBJECT TERMS Software Intelligent Agents, Software Management, Software Process Guide			15. NUMBER OF PAGES 117	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited

**DECISION SUPPORT FOR SOFTWARE PROCESS MANAGEMENT
TEAMS: AN INTELLIGENT SOFTWARE AGENT APPROACH**

Lori A. Church
B.S.E.E., San Diego State University, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

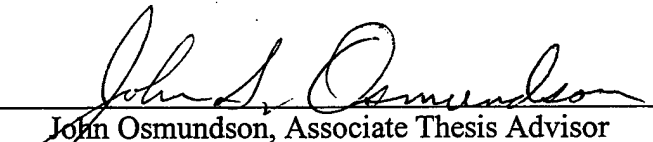
**NAVAL POSTGRADUATE SCHOOL
March 2000**


Author:

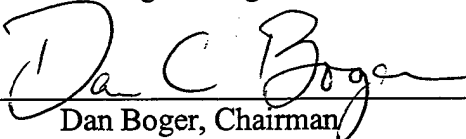

Lori A. Church

Approved by:


James Bret Michael, Thesis Advisor


John Osmundson, Associate Thesis Advisor


Luqi, Chairman
Software Engineering Curriculum


Dan Boger, Chairman
Department of Computer Science

ABSTRACT

Currently, SPAWAR Systems Center is lacking a unified software development environment that allows software developers to effectively manage software development projects across a diversified development environment. This unified environment is needed to provide up-to-date accurate information to the right people at the right time, increase the process knowledge-base, increase productivity, decrease time to market, eliminate redundancy, and ease job stress.

This thesis proposes a conceptual model for software process management decision support in the form of an intelligent software agent network. The intelligent software agent network, called MENTOR, provides the knowledge-base that is crucial to the software development team, providing for a repeatable, defined, managed, and optimized development environment. This concept provides SSC software development managers and team members with the ability to work in a unified and collaborative environment, regardless of organizational diversity or location.

MENTOR will be utilized as an integral software development team member, providing tutorials and mentoring capabilities for management and process assistance, as well as providing process planning, risk analysis, and strategic planning recommendations for the successful completion of a software development effort, at all team levels. In addition, MENTOR will provide an effective communication environment that will enable the development team to minimize the time consuming workload involved in tracking individual tasking.

TABLE OF CONTENTS

I.	INTRODUCTION AND BACKGROUND	1
A.	MOVTVATION	1
B.	PROBLEM STATEMENT	4
C.	RESEARCH SCOPE AND OBJECTIVES	5
D.	EXPECTED CONTRIBUTIONS	5
E.	METHODOLOGY	6
F.	THESIS OVERVIEW	6
II.	DESIGN FOUNDATIONS	9
A.	MENTOR DATABASE FRAMEWORK	9
1.	Organizational Process Asset Database (OPAD)	10
a.	Project Planning	13
b.	Requirements Definition	16
c.	Risk Management	19
d.	Configuration Management	23
e.	Quality Assurance	25
f.	Capability Maturity Model	26
g.	Estimation	30
h.	Lessons Learned	32
i.	Life Cycle Development Models	33
j.	Tracking and Oversight	36
k.	Training	38
l.	Tools	39
m.	Resource Library	40
2.	Project Process Asset Database (PPAD)	41
3.	Agent Rules and Knowledge Database (ARKD)	41
III.	MENTOR DESIGN GOALS	43
A.	INTERACTIVE PROJECT TUTORIAL	43
B.	PROJECT PROCESS MANAGEMENT	45
C.	LESSONS LEARNED	46
D.	STRATEGIC PLANNING	48
E.	USER INTERFACE	49
IV.	AGENT BACKGROUND	51
A.	WHAT IS AN INTELLIGENT AGENT?	51
B.	BASIC AGENT ARCHITECTURE	52
C.	AGENT CLASSIFICATION	53

1.	Intelligence.....	54
2.	Mobility	54
3.	Lifetime.....	55
4.	Interaction	55
5.	Task Specificity	56
6.	Behavior.....	56
7.	Environment.....	58
8.	Initiative	58
D.	AGENT ISSUES.....	59
1.	Language and Platform	59
2.	Control	59
3.	Resource Management.....	60
4.	Privacy	60
5.	Communication.....	60
6.	Mobility	60
7.	Understanding	62
E.	MENTOR'S DEVELOPMENT LIFE CYCLE.....	62
V.	MENTOR - A SOFTWARE AGENT CONCEPT FOR THE SOFTWARE MANAGEMENT PROCESS	65
A.	SYSTEM CONTEXT	65
B.	AGENT FUNCTIONALITY.....	67
C.	AGENT PROFILES AND BEHAVIORS	69
1.	Use Case 1 – Interact with Personal Assistant.....	69
2.	Use Case 2 – Setup User Interface.....	70
3.	Use Case 3 – Check Project Status/Alarms/Alerts	71
4.	Use Case 4 – Access Interactive Process Guide	72
5.	Use Case 5 – Access Interactive Strategic Planning.....	73
6.	Use Case 6 – Access Interactive Lessons Learned	74
7.	Use Case 7 – Access Interactive Tutorial	75
8.	Use Case 8 – Access Internet.....	76
9.	Use Case 9 – Access Applications.....	77
10.	Use Case 10, 11, and 12 – Maintain Agent Team, Coordinate Database Management, and Coordinate MENTOR System Attributes.....	78
11.	Project Process Asset Database Facilitator Agent	79
12.	Organizational Process Asset Database Facilitator Agent.....	80
13.	Agent Management Coordinator Agent.....	80
14.	Database Agents.....	81
15.	Task Specific Mobile Agents.....	82
D.	CONCEPTUAL MENTOR AGENT ARCHITECTURE	83
E.	CONCEPTUAL MENTOR SYSTEM ARCHITECTURE.....	85
VI.	SOFTWARE AGENT CASE SCENARIO	89

A.	TEAM BACKGROUND AND ASSUMPTIONS.....	89
B.	HYPOTHETICAL SOFTWARE PACKAGE.....	90
C.	MANUAL ESTIMATION BASELINE SCENARIO	91
D.	MENTOR ESTIMATION SCENARIO	94
VII.	SUMMARY AND RECOMMENDATIONS.....	101
VIII.	FUTURE WORK.....	103
A.	DETAILED ANALYSIS AND DESIGN.....	103
B.	FIRST PHASE SYSTEM IMPLEMENTATION	103
C.	SYSTEM SECURITY	104
D.	INTEGRATION OF OTHER SPAWAR THESIS EFFORTS INTO THE OPAD.....	104
E.	HIGH PERFORMANCE ORGANIZATION MODEL	105
F.	SEI CERTIFICATION	106
	LIST OF REFERENCES.....	107
	BIBLIOGRAPHY.....	111
	INITIAL DISTRIBUTION LIST	117

LIST OF ACRONYMS

AIF	Access Internet Facilitator
AMC	Agent Management Coordinator
ARKD	Agent Rule and Knowledge Database
CA	CMM Agent
CASE	Computer Aided Software Engineering
CCP	Credit Card Procedure
CMA	Configuration Management Agent
CMM	Capability Maturity Model
COCOMO	Constructive Cost Model
COTS	Commercial Off The Shelf
CPS	Check Project Status
CR	Change Request
EA	Estimation Agent
GOTS	Government Off The Shelf
HPO	High Performance Organization
I&DM	Information & Decision Management
ILLC	Interactive Lessons Learned Coordinator
IPGC	Interactive Process Guide Coordinator
ISPC	Interactive Strategic Planning Coordinator
ITC	Interactive Tutorial Coordinator
KPA	Key Process Area
LCDA	Life Cycle Development Agent
LLA	Lessons Learned Agent
LOC	Lines Of Code
MIL-STD	Military Standard
NOG	Naturally Occurring Group
OF	OPAD Facilitator
OPAD	Organizational Process Asset Database
OTA	Oversight and Tracking Agent
PCR	Process Change Recommendations
PDA	Project Database Agent
PF	PPAD Facilitator
PPA	Project Planning Agent
PPAD	Project Process Asset Database
PR	Problem Report
QA	Quality Assurance
QMA	Quality Management Agent
RA	Resources Agent
RAF	Risk Assessment Form
RDA	Requirements Definition Agent
REVIC	Revised Intermediate COCOMO

RM	Risk Magnitude
RMA	Risk Management Agent
RP	Remote Programming
RPC	Remote Procedure Call
SA	Specific Applications
SDD	Software Design Document
SDF	Software Development File
SDL	Software Development Library
SDP	Software Development Plan
SE	Software Engineering
SEF	Software Engineering File
SEI	Software Engineering Institute
SEPO	Software Engineering Process Office
SM	System Maintenance
SOW	Statement Of Work
SPAWAR	Space and Naval Warfare Systems Center
SPE	Software Project Engineering
SPM	Software Program Manager
SPP	Software Project Planning
SPTO	Software Project Tracking and Oversight
SQA	Software Quality Assurance
SRS	Software Requirements Specification
SSC	SPAWAR Systems Center
SSDD	System Software Design Document
SSM	Software Subcontractor Management
SSS	System Software Specification
SUI	Setup User Interface
SW-CMM	Software Capability Maturity Model
TA	Training Agent
TRA	Tools Resource Agent
TSM	Task Specific Mobile
UML	Unified Modeling Language
UPA	User Personal Assistant
WWW	World Wide Web

ACKNOWLEDGEMENT

It is with great appreciation that I would like to thank Dr. James Bret Michael for his guidance and support, but most of all his patience and Dr. John Osmundson for his guidance and inspiration that lead me to develop this thesis topic. I would also like to thank my family and friends for supporting me throughout this effort. And finally, I offer a heartfelt thanks to my husband and parents, for without their encouragement and support I would not be writing this today.

I. INTRODUCTION AND BACKGROUND

A. MOVTVATION

More and more, we are dependent upon software to complete our daily tasks as software managers and developers. We are also being asked to perform more work with less. We look for ways to increase productivity, ease job stress, and develop software faster and more efficiently than ever before. We have to be smarter and faster in how we do business and develop software. In order to do this, we must increase our knowledge base and be able to access the right knowledge at the right time. The information our knowledge is based on must also be as up-to-date and accurate as possible. Bill Gates states in his new book, *Business@The Speed of Thought*, "How you gather, manage, and use information will determine whether you win or lose." [Ref. 7]

It is this availability of information that Gates calls a "digital nervous system". The information provided by the digital nervous system is needed in varying degrees at all levels of an organization. When the right information is available to those that need it, the information has the most impact, and therefore, there are more opportunities to provide input and innovative ideas to the company. Gates writes, "... still another sign of a good digital nervous system is the number of good ideas bubbling up from your line managers and knowledge workers." [Ref. 7]

The knowledge that is crucial to the software development team is specific knowledge that will provide for a repeatable, defined, managed, and optimized development environment. Currently, much of the knowledge responsibility resides with the program manager who needs expertise in all aspects of the development process. A

program manager must know the process from cradle to grave, that is, he must be an expert in all aspects of planning, risk management, engineering, tracking and oversight, as well as be an excellent customer interface agent and guide the team to the successful completion of the development effort.

The Software Engineering Institute (SEI), at Carnegie Mellon University, recognized that there is a common myth that the major problems occurring in software development projects are technical. But in fact, they are managerial. This is backed by SEI assessment and evaluations, as well as the Defense Science Board Task Force Report on Military Software, 1987. [Ref. 11]

SEI is trying to put this myth to rest. As many companies are trying to achieve software certification through the Software Engineering Institute (SEI), the information, knowledge, and coordination that is required to achieve even a Capability Maturity Model (CMM) Level 2 Certification can be quite demanding. If the entire team, not just the program manager, had expert information available at their fingertips to guide and mentor the team and organization through the development process using the CMM, better rates of success in software development projects could be achieved.

Imagine an assistant that knew every detail of the software development process. What if the assistant could alert you to risks facing your project? What if it could perform strategic thinking and projections? What if it could mentor your software development team and guide you through the SEI certification process and help you achieve the highest level possible? What if it could find the information that you and your team need at the drop of a hat? What if the assistant could keep all the historical data and lessons learned from previous projects to provide a basis for planning and

estimation? This would allow for a proactive development environment instead of the current reactive one. Risks could be mitigated or even eliminated before they became problems. The assistant would learn from the past teams' mistakes and take them into account the next time the same situation arises. The assistant would remind the team member of scheduling constraints and keep him or her on track throughout every step of the process. Now, what if the assistant is an intelligent agent instead of a real person?

Intelligent agents are being used increasingly in the areas of military strategic planning, scheduling, and inventory control, as well as increasing Internet applications for wizards to facilitate browsing. They are used in fields such as robotics, intelligent and adaptive interfaces, intelligent search and filtering on the web, and information retrieval, just to name a few. Over the years we have developed software tools that can re-engineer software to create flowcharts, track requirements, develop software test cases automatically, and reuse existing software, all in the name of easing the workload of the software developer. Intelligent agents are helping us realize these services.

These goals are being achieved through a multitude of automated tools, utilizing agents available for the software process, but their participation is very development oriented. What is needed is an assistant that will tutor the team members in the development process. Whether team members are new or experienced, the agent would guide them through the development process in an actual program, gather lessons learned from past projects providing insight into planning and estimation, and offer strategic planning solutions and projections for the successful completion of software projects.

A network of intelligent agents that mentor, new and experienced, managers and developers could greatly benefit Space and Naval Warfare Systems Center (SSC) and

their vision for making Information and Decision Management (I&DM) a reality. SSC is dedicated to giving “the right people the right information at the right time” and to “help integrate disparate groups and functions into coordinated operations.” [Ref. 23]

Currently, SSC provides an I&DM capability for emergency, disaster preparedness, and crisis management projects. This I&DM vision and technology, used in emergencies, should also be applied to the management of software projects.

Currently, there are many different groups that develop software at SSC. Each group independently provides planning, risk management, and engineering. The network of agents, which has been named MENTOR, will provide a joint resource for collecting historical data, estimating, planning, and risk mitigation. MENTOR will be available via the SSC Intranet and Internet, which will allow the entire SSC development team to work together in a unified and collaborative environment, regardless of organizational diversity or location.

MENTOR will be utilized as an integral software development team member in providing process assistance, tutorials and mentoring capabilities for management. In addition, MENTOR will provide process planning, risk analysis, and strategic planning recommendations for the successful completion of a software development effort.

B. PROBLEM STATEMENT

A unified software development environment, that assists the software developers in managing software development projects and tasking across a heterogeneous development environment, is currently unavailable. The consequences of such a state of affairs has resulted in the inability to:

- Provide up-to-date accurate information to the right people at the right time
- Increase the process knowledge base
- Increase productivity
- Decrease time-to-market
- Eliminate redundancy
- Ease job stress and task workload

C. RESEARCH SCOPE AND OBJECTIVES

The research objective is to develop a conceptual top-level software engineering design for the application and utilization of software agents that provide automated decision-making capabilities, tutorial guides, process mentoring, and both strategic planning and projection support for software development team members. Research for this thesis included investigation of the following design goals:

- Interactive Project Management Tutorial
- Project Management Process Guide
- Lessons Learned
- Strategic Planning and Projections

Team profiles, context diagrams, and use cases are used to develop the conceptual design for MENTOR. A conceptual agent architecture is proposed by identifying the types of agents needed and the behaviors the agents possess. Finally, a case study consisting of agent role playing scenarios from the estimation phase of the development process will be used to show the feasibility of the MENTOR concept.

D. EXPECTED CONTRIBUTIONS

This research will bring a much-needed unified software development environment to SSC San Diego that provides the following services:

- Provide up-to-date accurate information to the right people at the right time
- Increase the process knowledge-base
- Increase productivity
- Decrease time-to-market
- Eliminate redundancy
- Ease job stress and task workload

MENTOR will guide a software development manager and team members through the process, providing increased insight to make crucial management decisions necessary to complete projects on time and within budget. It will provide lessons learned for planning and estimation, and the desired strategic planning and projections. But most of all, it will provide the basis for the crucial information flow and digital nervous system that is needed to allow the right people at the right time to get the information in a fast paced, cutting-edge, software development environment.

E. METHODOLOGY

An investigation and review of the literature to formulate an overview of current agent technology and background concepts was conducted, as well as the background information needed to develop the conceptual process management environment in which agents work.

Based on this research, a conceptual model for MENTOR was developed. A case scenario was used to validate the concepts and feasibility of the MENTOR model. The case study includes scenarios from the estimation phase of the MENTOR process model.

F. THESIS OVERVIEW

Design Foundations in Chapter II of this thesis provides an overview of the MENTOR database framework and the information assets which reside in the database

that are essential to the success of this intelligent agent network. Chapter III focuses on design goals that provide the underlying structure and functionality for MENTOR. Chapter IV provides background information on agents, their basic architecture, a means of classification, issues of concern, and an overview of an agent's development life cycle. Chapter V addresses the conceptual approach for the MENTOR process management agent network. It outlines the system context and provides use cases for top level MENTOR functionality. The use cases are then implemented conceptually through the use of software agent team profiles. A conceptual agent architecture is then proposed based on the context diagrams, use cases, and team profiles. Chapter VI describes an example environment for software development. A hypothetical software package development effort is used to compare a baseline manual method scenario to a MENTOR method scenario in order to show basic information flow and feasibility of the MENTOR agent architecture for the estimation process. Chapter VII offers a summation of thesis efforts and recommendation for SPAWAR to continue the development process of MENTOR. Chapter VIII discusses future work possibilities for MENTOR. These include detailed analysis and system design, a first phase implementation approach, system security considerations, incorporation of other SPAWAR thesis efforts, Software Engineering Institute (SEI) Certification, and High Performance Organizational Model implementation.

THIS PAGE INTENTIONALLY LEFT BLANK

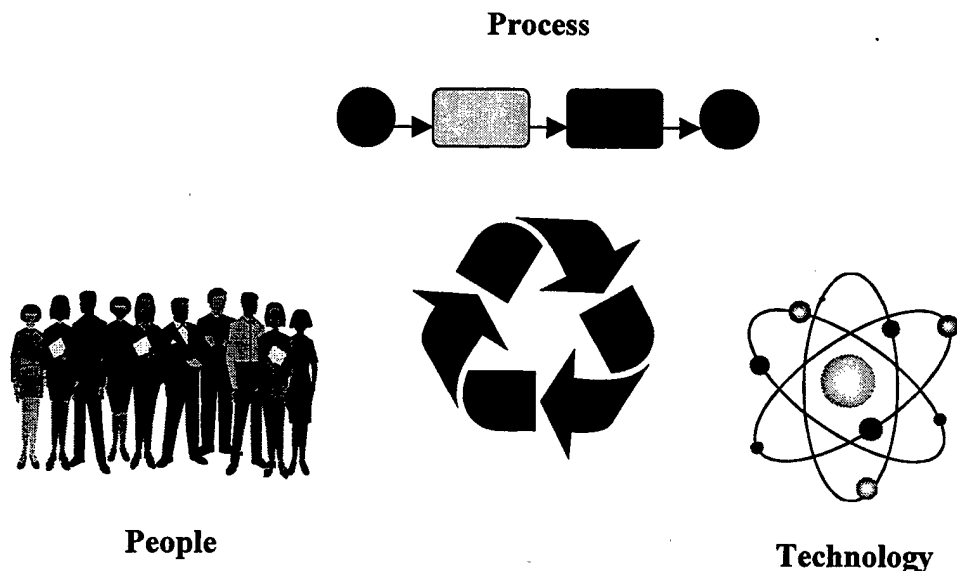
II. DESIGN FOUNDATIONS

MENTOR is an integrated agent network that will provide software managers and their team members with a unified project environment that will enable the entire team to produce high-quality software products. In order to accomplish this, a framework needs to be developed in which the MENTOR agents can interact. This framework is outlined in this chapter:

A. MENTOR DATABASE FRAMEWORK

MENTOR will bring together the three elements needed for project success:

Process, People, and Technology as shown in Figure 2.1.



**Improved Process + Competent Workforce + Appropriate Technology =
Reduced Risk, Higher Productivity, and Better Quality**

Figure 2.1. Three Elements of Project Success After [Ref. 17]

In building the MENTOR concept, the first task was to define what is meant by *process*. Pressman's definition of a software process is "a framework for the tasks that are required to build high-quality software." [Ref. 14]. SEI's definition of process is "The means by which people, procedures, methods, equipment, and tools are integrated to produce a desired end result." [Ref. 11]

The MENTOR process combines both the Pressman and SEI definitions, with a management flair, into the *management of the framework by which people, procedures, methods, equipment, lessons learned, and tools are integrated to produce the high-quality environment needed to produces high-quality software.*

This process would not be possible without the appropriate technology that supports the entire framework. This technology encompasses all the hardware, software, and tools needed to allow the people to successfully implement and improve on the process. Part of this technology consists of databases providing the assets and knowledge that the agent network will utilize to support the people working on the project.

With this in mind, the conceptual framework, shown in Figure 2.2, was developed. It consists of an Organizational Process Asset Database (OPAD), Project Process Asset Database (PPAD), and Agent Rules and Knowledge Database (ARKD). The following sections will outline what is contained in each of the databases.

1. Organizational Process Asset Database (OPAD)

The OPAD is a common organizational data repository providing part of the underlying framework for the MENTOR unified software development environment. Support and information for the following 13 key knowledge-base areas are as follows:

- Project Planning
- Requirements Definition
- Risk Management
- Configuration Management
- Quality Assurance
- Capability Maturity Model
- Estimation
- Lessons Learned
- Life Cycle Development Models
- Oversight and Tracking
- Training
- Tools
- Resources Library

Information in the 13 key areas will be gathered from individual projects throughout SSC and the vast Software Engineering Process Office (SEPO) resource library. This will allow for a common repository of data, processes, and lessons learned helping to eliminate the current repetitive nature and reinvention of the wheel processes and information that go on constantly throughout SSC. The PPAD contains all current project artifacts that are being developed. Once the artifacts have been approved for distribution, they are moved to the OPAD for organization-wide use. The Agent Rules represent the portion of MENTOR that consists of the agent network that guides users through the life cycle process. Starting with project planning, the assets contained in each will be outlined.

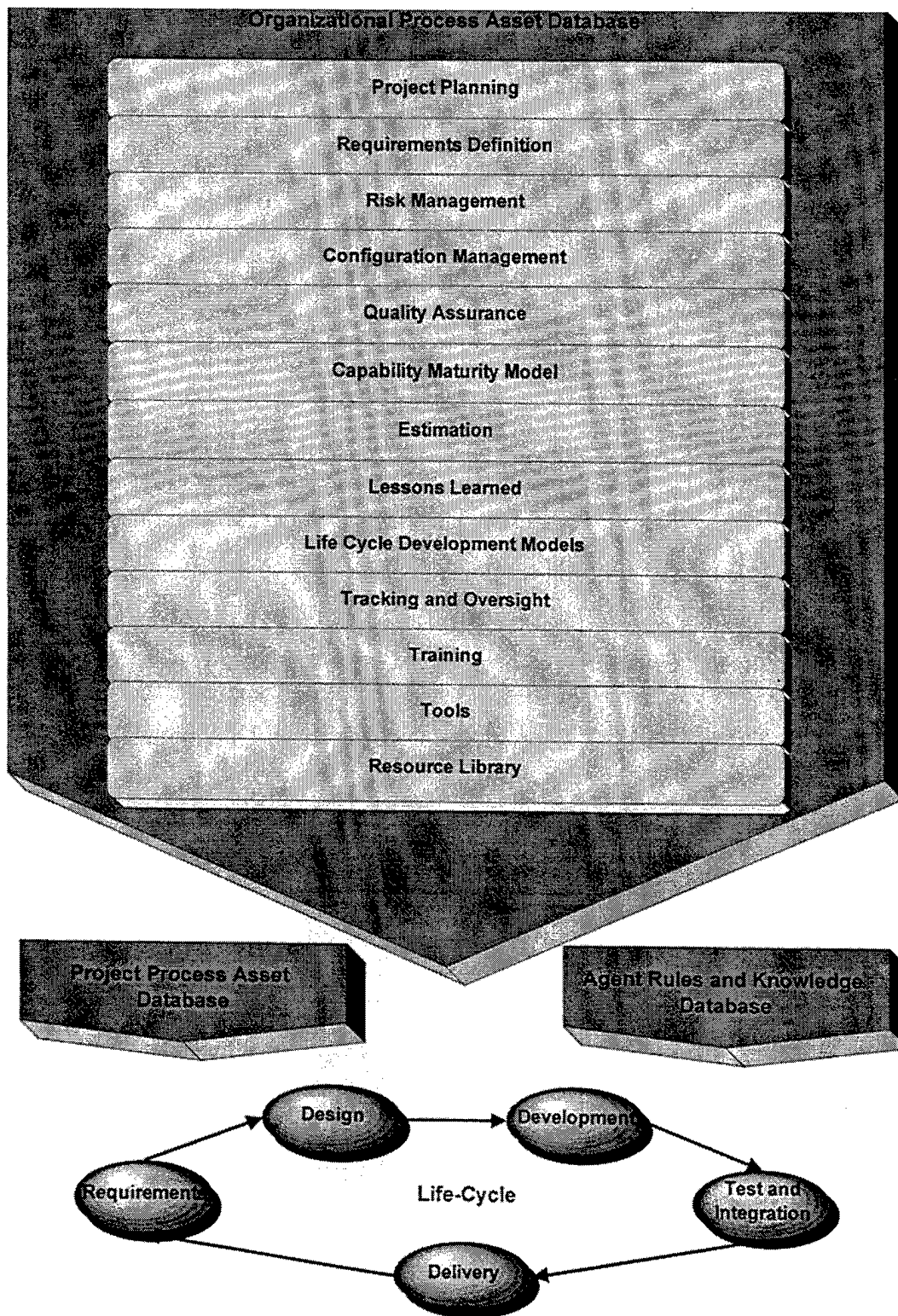


Figure 2.2. Conceptual MENTOR Framework

a. Project Planning

Project planning is one of the most important steps in a software development effort which allows the program manager and team to identify tradeoffs, risks, resources, and products early in the project life cycle. As the SEPO states in the their Software Project Planning Process document, "lack of adequate planning often results in a project's failure to meet either cost, schedule, or performance objectives or all three." [Ref. 18] The main goals for project planning from the SEPO Software Management for Executives Guidebook are:

- Software estimates are documented for use in planning and tracking the software project.
- Software activities and commitments are planned and documented.
- Affected groups and individuals agree to their commitments relating to the software project.

Achievement of the preceding goals will aid in the establishment of "reasonable plans for performing the software engineering and for managing the software project." [Ref. 16] It also allows visibility of how the project is being managed (defining what the work is and how it will be done), as well as describing the procedures for managing the project. The plan for reaching the guidebook goals is documented in the Software Development Plan (SDP). The SDP includes information and plans pertaining to project tracking, risks, schedule, cost, size, resources, methodologies, and technologies to be used during development. Mentor will guide the program manager in the development of this document and then use the same information to mentor the project manager to the successful completion of the software development project.

Since the SDP is a living document that "guides the software project manager and staff members through the software development process," the planning

process extends from the beginning of a project to its completion. [Ref. 16] The planning process that MENTOR will follow, as outlined in the SEPO Software Project Planning Process document, is shown in Figure 2.3, followed by a planning process legend in Table 2.1.

Resource documentation for this process will include the SSC Planning Policies and the following templates and samples thereof:

- Software Development Plan
- Software Development File
- Software Development Library
- Software Transition Plan

MENTOR will utilize these resources to help the project manager create a comprehensive plan to ensure that the following critical factors are met:

- Defines project schedule and budgetary goals
- Defines areas of responsibility
- Schedules for high-level tasks down to greater levels of detail
- Establishes task sequences
- Defines Major/Minor Milestones
- Assigns resources to tasks
- Calculates project budget on a task-by-task basis

Through the use of tools like MS Project, MENTOR will produce Activity Networks, Gantt Charts, calendars, work-hour forms, and status reports for planning activities. MENTOR will also provide an adaptable Project Process checklist for all development team levels, which can be used to track progress. This will ensure that process steps are not missed.

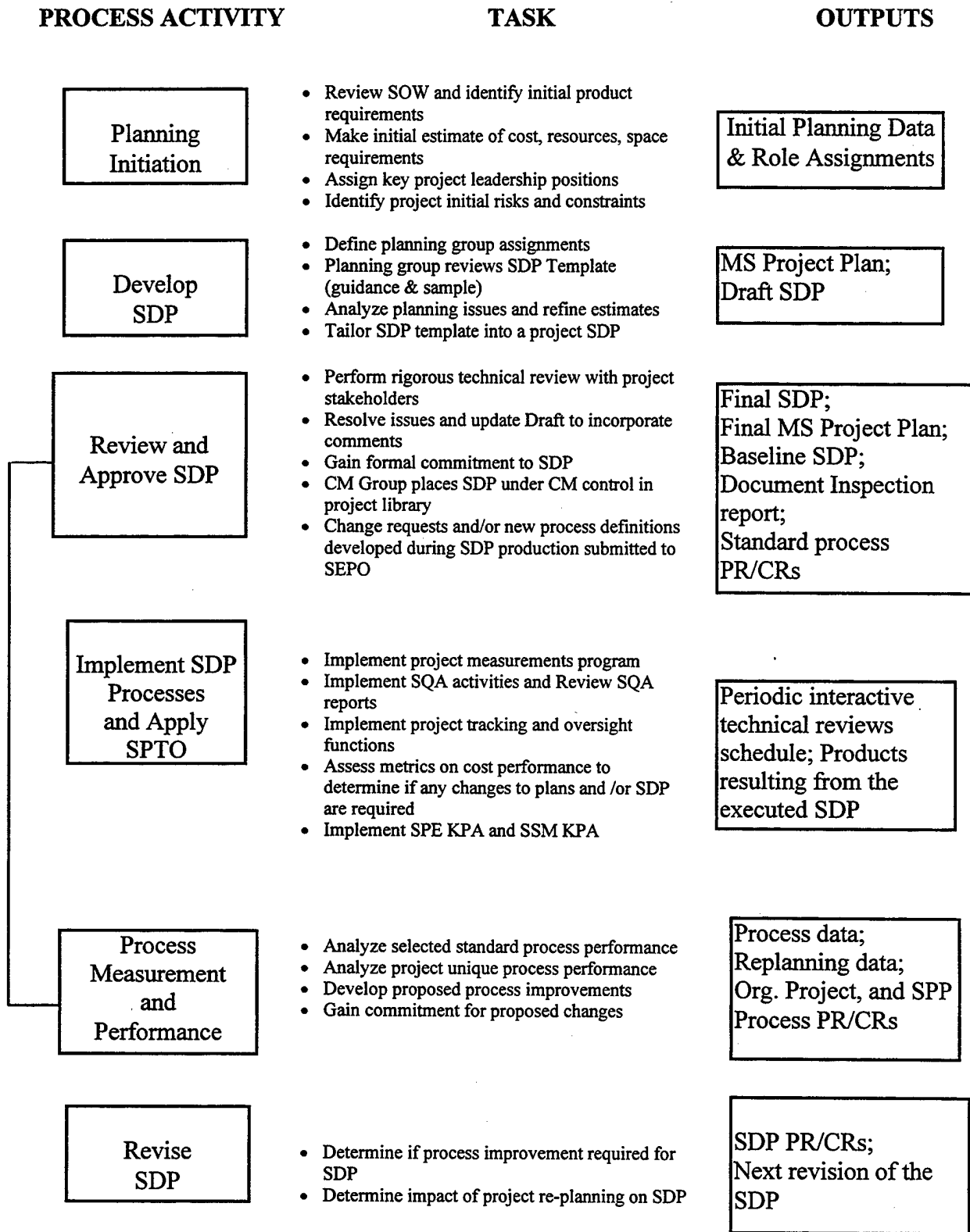


Figure 2.3. Project Planning Process From [Ref. 16]

Project Planning Process Legend	
CM – Configuration Management	SOW – Statement Of Work
CR – Change Report	SPE – Software Project Engineering
KPA – Key Process Area	SPP – Software Project Planning
PR – Problem Report	SPTO – Software Project Tracking and Oversight
SDP – Software Development Document	SQA – Software Quality Assurance
SEPO – Software Engineering Process Office	SSM – Software Subcontractor Management

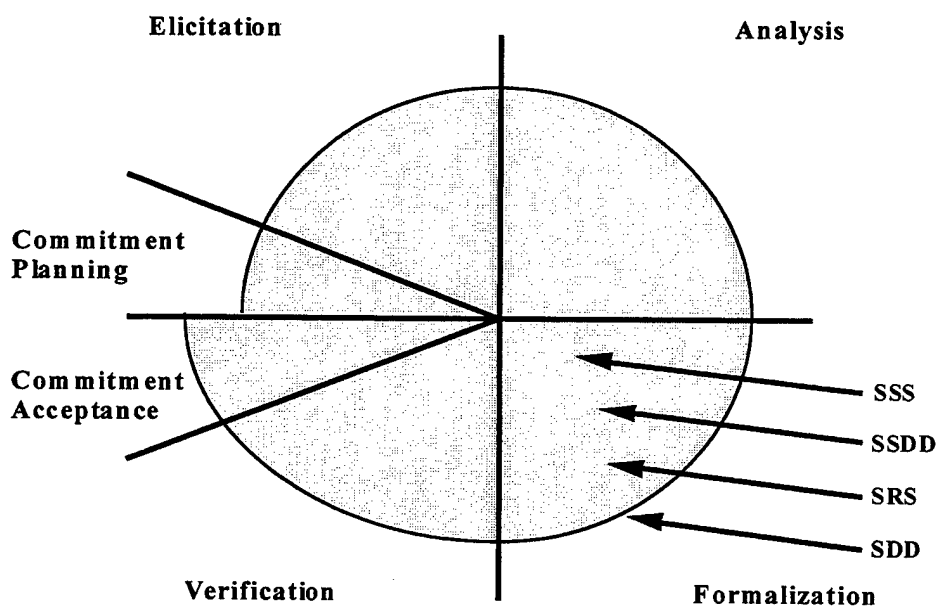
Table 2.1. Legend for Project Planning Process (Figure 2.3)

b. Requirements Definition

Requirements Definition is one of the most important considerations in the software development process. Brooks writes in his book *The Mythical Man-Month* that “The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is so difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.” [Ref. 4] These difficulties result in statistics, such as, 53% of all software projects cost nearly 90% over the original estimates, 42% of the original proposed features and functions are implemented in the final product, 31% of all software projects are cancelled prior to final delivery, 40% of a software projects budget is wrapped up in rework, and 70% of all rework is caused due to inadequate requirements definition at the beginning of the project. [Ref. 17] Therefore, it is important that the software development team fully understands what the customers want and what they need developed. This understanding is accomplished by providing the knowledge and the tools needed to provide clarity of requirements.

The requirements definition database will contain all the information by the MENTOR agent network to guide the development team through the requirements definition process. Much of the information is already developed and can be found in document form via the SEPO WWW Homepage. The Requirements Management Guidebook is just one of the sources and provides the basic framework and process model for requirements management, as shown in Figure 2.4.

The database will provide guiding information, through MENTOR agents, that will aid the project manager in identifying and clarifying the participants, entry criteria, input, steps, exit criteria, and output for each activity in the requirements management process model.



Requirements Definition Process Legend

SDD – Software Design Document
SRS – Software Requirements Document

SSDD – System Software Design Document
SSS – System Software Specification

Figure 2.4. Requirements Definition Process From [Ref. 15]

This database will contain a requirements management checklist that is maintained by a MENTOR agent via team member inputs and suggestions for metric collection. It will provide outlines for required inputs, participants, activities, products, and processes of requirements management. Government standards, the SSC San Diego Requirements Management Policy, document samples, templates and other pertinent reference documentation that govern requirements management activities can be tailored to meet the needs of the development team. MENTOR will also be able to access lists of terms, definitions, roles, and responsibilities needed for requirements definition from this repository.

Once the project managers have tailored the requirements process for the team's specific project, the tailored processes and documents will be saved in the Project Process Asset Database (PPAD) for ongoing project management of each specific project. MENTOR will then gather metrics using an off-the-shelf requirements-management tool for submission to the PPAD.

Through the use of this, MENTOR enables the project manager and development team to reduce the risk of cost and schedule slips, rework costs, requirements changes, and late program requirements errors.

MENTOR will also provide a customizable Requirements Management checklist for all development team levels, which can be used to track progress. This will ensure that process steps are not missed.

c. Risk Management

Risk Management allows the project manager and development team to discover potential problem areas as early as possible in the development cycle in order to take a proactive stance in the mitigation, reduction, or avoidance of risks. The risk management process that the SEPO has developed is shown in Figure 2.5. This process will enable MENTOR to help the project manager and development team identify, analyze, plan, track, control, communicate, and document risks related to software development effort.

The risk management database will include templates and samples, which can be tailored to specific project needs, including definitions, policies and references for risk management techniques. At present, risk management references can be found on the SEPO WWW Homepage <http://sepo.spawar.navy.mil>.

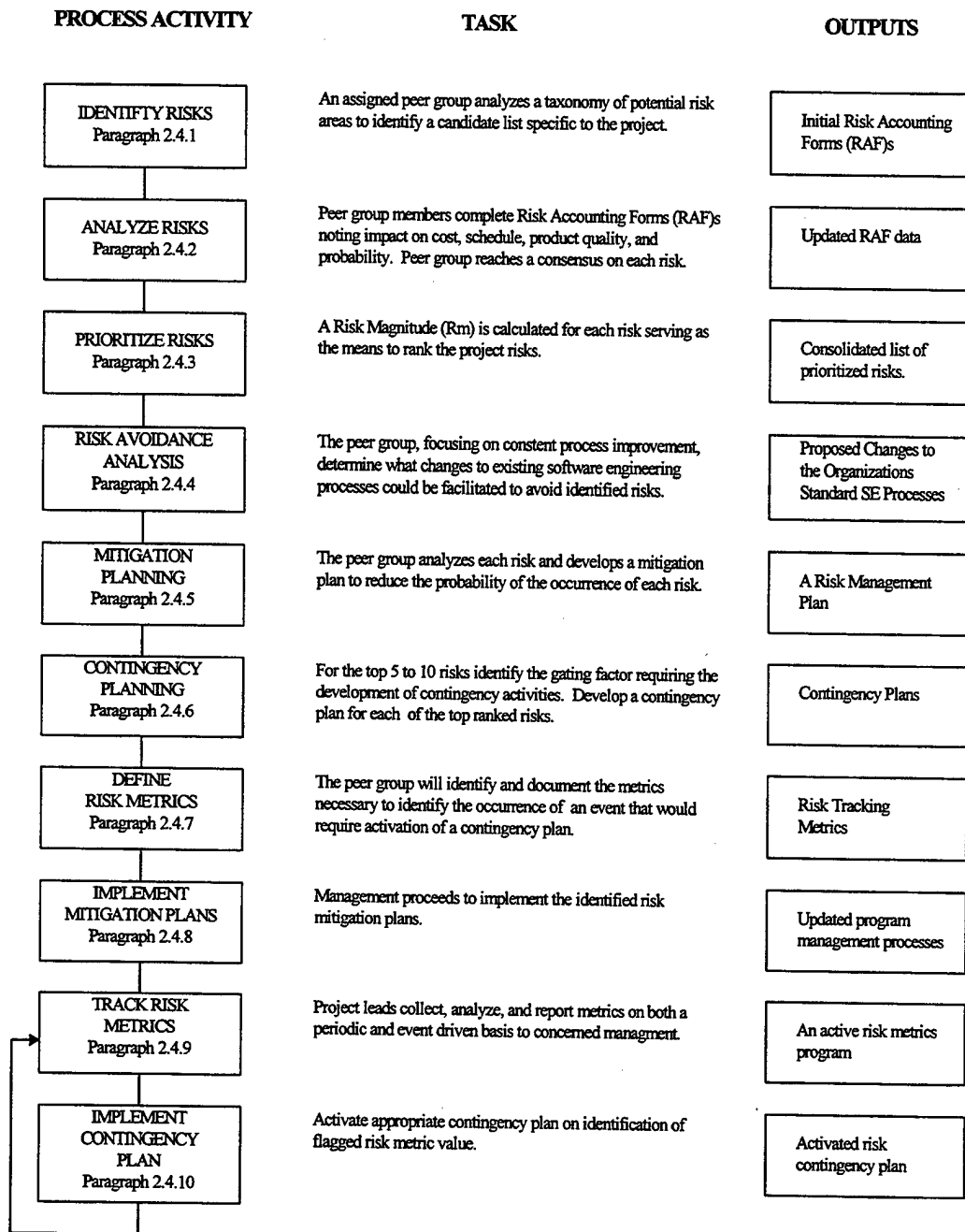


Figure 2.5. Risk Management Process Overview From [Ref. 19]

MENTOR will be able to walk the project manager and team through the risk identification process by utilizing risk lists such as the ones shown in Table 2.3.

Potential Software Development Risks		
A. Product Engineering	B. Development Environment	C. Program Constraints
1. Requirements	1. Development Process	1. Resources
a. Stability	a. Formality	a. Schedule
b. Completeness	b. Suitability	b. Staff
c. Clarity	c. Process Control	c. Budget
d. Validity	d. Familiarity	d. Facilities
e. Feasibility	e. Product Control	2. Contract
f. Precedent	2. Development System	a. Type of Contract
g. Scale	a. Capacity	b. Restrictions
2. Design	b. Suitability	c. Dependencies
a. Functionality	c. Usability	3. Program Interfaces
b. Difficulty	d. Familiarity	a. Customer
c. Interfaces	e. Reliability	b. Associate Contractors
d. Performance	f. System Support	c. Subcontractors
e. Testability	g. Deliverability	d. Prime Contractors
f. Hardware Constraints	3. Management Process	e. Corporate Management
g. Non-Developmental	a. Planning	f. Vendors
3. Code & Unit Test	b. Project Organization	g. Politics
a. Feasibility	c. Management Experience	
b. Testing	d. Program Interfaces	
c. Coding/Implementation	4. Management Methods	
4. Integration & Test	a. Monitoring	
a. Environment	b. Personnel Management	
b. Product	c. Quality Assurance	
c. System	d. Configuration Management	
5. Engineering Specialties	5. Work Environment	
a. Maintainability	a. Quality Attitude	
b. Reliability	b. Cooperation	
c. Safety	c. Communication	
d. Security	d. Morale	
e. Human Factors		
f. Specifications		

Table 2.3. Potential Software Development Risks After [Ref. 19]

The risk management database will provide a resource for a vast database of risks that have been identified on other projects, as well as potential solutions through

mitigation, reduction, and contingency planning. The database will identify risks according to identification fields shown in Table 2.4, as identified by the SEPO.

Database Risk Identification Fields	
Field Name	Field Description
ID Number	Unique identifier having specific project and risk number characteristics
Risk Name	Short phrase by which the risk will be known
Risk Description	Short description of what the risk is, its makeup and components
Reasons/Rationale for Probability	Justification and explanation of circumstances and past events that imply a degree of likelihood/probability for the risk
Probability	Assessment of the risk's probability of occurrence (very high, high, medium, and low)
Origin Date	Date the risk is first put into the database
Name POA	Name of the person who is the Point of Action for managing the risk
Category	Area that will be impacted by risk occurrence
Impact Statement/Rationale	Assessment description of the impact if the risk occurs
Impact Value	Assessment of the risk's severity of impact (critical, high, medium, and low). It is estimated based upon the rationale above.
Priority	Product of the probability and impact value yielding red for high risks, yellow for medium risks, and green for low risks
Time Frame	Estimate of the calendar time for which this risk exists or applies
Risk Avoidance	Description of an approach that completely eliminates/avoids the risk
Risk Reduction	Ways that the risk can be mitigated or its likelihood of occurrence being reduced.
Risk Reduction Triggers	Criteria for implementing/initiating a specific risk reduction technique
Indicators/ Metrics	Indicators or measurements that will be collected to track the risk.
Indicators/ Metrics Source	Source or place from which the indicators or measurements will be extracted

Table 2.4. Risk Management Database Field Descriptions After [Ref. 19]

Database Risk Identification Fields	
Field Name	Field Description
Indicators/ Metrics Frequency	Frequency of indicator or measurement is collected
Contingency	Description of what will be done to minimize the impact when the actual risk happens
Contingency Trigger	Criteria for implementing/initiating a specific risk contingency
Risk Status	Description of where the risk stands in its life cycle, what risk reduction approaches are in place, etc.

Table 2.4 Continued. Risk Management Database Field Descriptions After [Ref. 19]

The Risk Magnitude Matrix, shown in Table 2.5, provides a means for identifying and prioritizing risks.

Risk Magnitude Matrix			
Severity of Impact	Very Likely > 75%	Likely as Not 50-50	Unlikely < 25%
Critical			Yellow
Serious		Yellow	
Satisfactory	Yellow		

Risk Magnitude = Severity of Impact * Probability of Occurrence
Risk Magnitude = Priority of Importance * Likelihood of Happening

Table 2.5. Risk Magnitude Matrix After [Ref. 19]

MENTOR will also provide a configurable Risk Management checklist for all development team levels, which can be used to track progress. This will ensure that process steps are not missed.

d. Configuration Management

Configuration management establishes and maintains integrity of the products developed during the life cycle of the software development effort and is a "set of activities developed to manage change throughout the software life cycle." [Ref. 9] It

is the process by which software elements, such as source code and the corresponding documents are baselined, controlled, and updated in a defined and repeatable manner.

CM is carried out over the entire life cycle of the software project. It plays four distinct roles in the software development process:

- CM Audits - ensures the system provides the expected and required deliverable integrity
- Status Accounting – informs stakeholders of the status of baselines and proposed changes to those baselines
- Control – establishes baselines and controls changes made to those baselines
- Identification – uniquely identifies key deliverables and support of configuration items

CM will provide resources for identifying configuration items, performing CM audits, recognizing what is included in status accounting, and noting what items need to be controlled. It will also serve as a CM repository to hold the Software Development File (SDF) and Software Development Library (SDL).

The SDF is a repository for collecting material pertinent to the software life cycle and development effort. Typical items found in the SDF are the following:

- Design considerations
- Design constraints
- Major coding considerations
- Test information
- Schedule and status information

The SDL is a controlled library of software documentation and all configuration items and any other data that is pertinent to the project at an organizational level.

Guidelines for performing baseline functions, when to baseline, the approval process, unit testing configurations, and procedures for documenting code are

identified as resources. Resources, procedures, and tools will also be provided to establish levels of control, provide types of reviews to be performed and applied at each level, provide interface controls within the product and across product boundaries and with the environment, and identify items that should be controlled. Also included are the following:

- Organization Software Configuration Management Policy
- Organization Software Configuration Management Processes
- Configuration Management Procedures
- Generic Software Configuration Management Plan
- Configuration Status Accounting Reports
- Sample Software Configuration Management Desktop Procedures
- Software Configuration Desktop Tool Selection Procedures

MENTOR will also provide a customizable Configuration Management checklist for all development team levels, which can be used to track progress. This will ensure that process steps are not missed.

e. Quality Assurance

Software Quality Assurance (SQA) consists of the methods and procedures that ensure software products will meet the customer's needs and stated requirements. The objectives of SQA are to: [Ref. 9]

- Improve software quality by monitoring both the software product and the software development process that produces it.
- Ensure full compliance with the standards and procedures identified for the software product and the software process.
- Ensure that discrepancies in the product, process, or standards are identified and resolved.
- Assist in the collection of process data to be fed back to the projects and the process group for continuous process improvement.

SQA is an overarching activity that spans the entire life cycle of the software development effort and support process improvement. The following summary

of activities was provided in the SSC Software Program Management Course. In the planning stage, SQA is used to:

- Identify appropriate standards, procedures, and tools
- Document quality roles and responsibilities
- Establish plans for performing quality functions
- Establish an appropriate development methodology

During the engineering stage, SQA is used to:

- Track product and process quality
- Ensure adherence to established standards
- Monitor project progress independently
- Foster use of best practices and teamwork

The SQA knowledge base will include all processes, procedures, guidelines, and resources in support of SQA, such as:

- SSC Software Quality Assurance Policy
- Software Quality Assurance Process
- Software Quality Assurance Plan Templates
- Software Quality Assurance Plan Samples

MENTOR will also provide an adaptable Quality Assurance (QA) checklist for all development team levels, which can be used to track the QA process, ensuring that process steps are not missed.

f. Capability Maturity Model

The Software CMM, as defined by SEI, is "A common-sense application of process management and quality improvement concepts to software development and maintenance." [Ref. 11] It was developed by Carnegie Mellon University under U.S. Air Force sponsorship and originally used to evaluate software contractor capabilities. As

stated in Phillips' *The Software Project Manager's Handbook*, it "evolved first into a process maturity framework and then into the CMM in its present form." [Ref. 12]

The CMM is a model that offers guidance for improvement to the organization for developing software that is repeatable, defined, managed, and optimized. [Ref. 11] However, Phillips recognized that many think the CMM requires too much documentation and seems to have lost some of its appeal since the early 90's. However, the benefits far outweigh the time required to develop the proper documentation. "The CMM also teaches that organizations with mature processes produce better software consistently." [Ref. 12]

The comprehensive underlying structure of the CMM provides a maturity level grading system for measurement of a software developer's engineering practices.

The Five Level Maturity Framework is represented in Figure 2.6.

- Level 1: Initial - The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.
- Level 2: Repeatable - Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
- Level 3: Defined - The software process for both management and engineering activities is documented, standardized, and integrated into an organization-wide software process. All projects use a documented and approved version of the organization's process for developing and maintaining software.
- Level 4: Managed - Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures.
- Level 5: Optimizing - Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies.

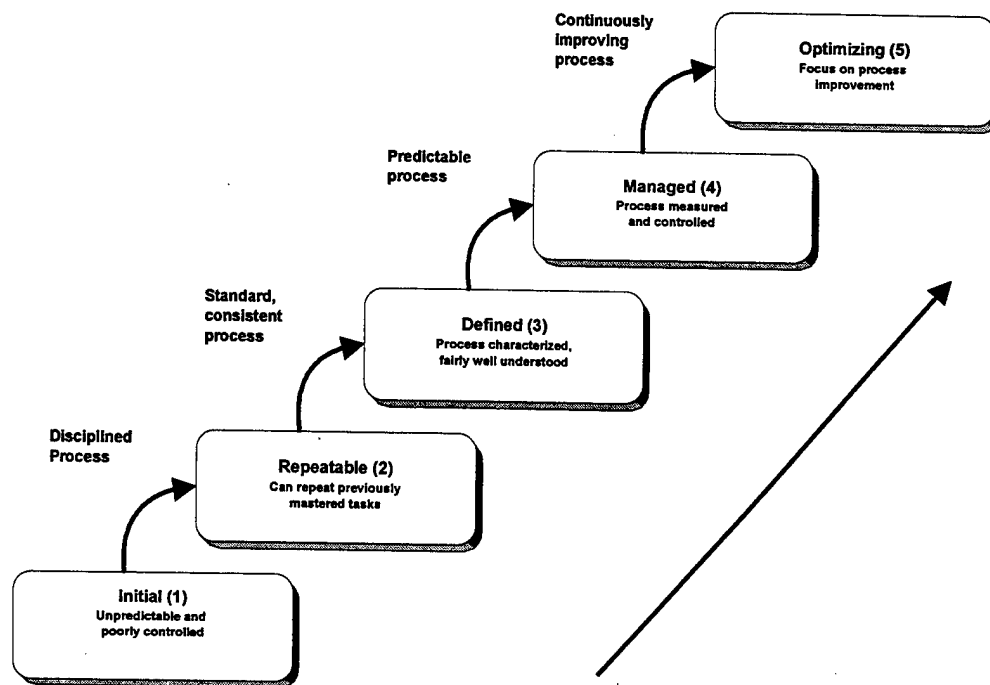


Figure 2.6. CMM Five Level Maturity Framework [Ref. 10]

SEI has developed key process areas (KPA) with each maturity level.

The KPAs describe the software engineering functions that must be satisfied at each level. Goals are set to achieve the KPAs. An overview of the CMM structure is provided in Figure 2.7. For the purposes of the MENTOR concept, the CMM will be visible to the KPA level and is defined in the SW-CMM v1.1, by SEI as in Table 2.5.

The description of the CMM is provided as an overview only and is included to familiarize the reader with the CMM concept. The CMM, in matrix form as given by SSC Software Engineering Process Office (SEPO), can be found in Appendix A with the SEPO's coverage the KPAs. The CMM, Version 1.1, in its entirety, can be found on the SEI WWW Homepage at <http://sei.cmu.edu>. The CMM is currently moving

to Version 2B, but is not published at this time. The SPAWAR SEPO WWW Homepage also provides a vast resource for the software process, including the CMM, and can be found at <http://sepo.spawar.navy.mil>.

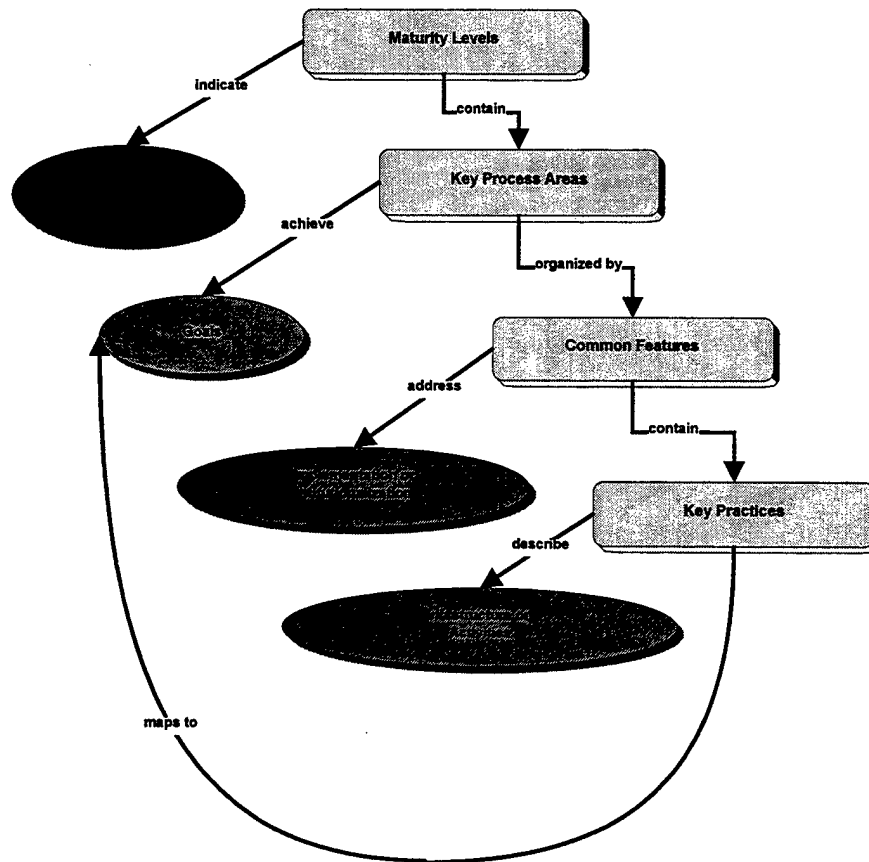


Figure 2.7. CMM Structure After [Ref. 11]

SW-CMM V1.1 KEY PROCESS AREAS (KPAS)		
Level	Focus	Key Process Areas
5 Optimizing	Continual Process Improvement	KPA18 – Process Change Management KPA17 – Technology Change Management KPA16 – Defect Prevention
4 Quantitatively Managed	Product and Process Quality	KPA15 – Software Quality Management KPA14 – Quantitative Process Management
3 Defined	Engineering Processes and Organizational Support	KPA13 – Peer Reviews KPA12 – Intergroup Coordination KPA11 – Software Product Engineering KPA10 – Integrated Software Management KPA9 - Training Program KPA8 - Organization Process Definition KPA7 - Organization Process Focus
2 Repeatable	Project Management Processes	KPA6 - Software Configuration Management KPA5 - Software Quality Assurance KPA4 - Software Subcontractor Management KPA3 - Software Project Tracking and Oversight KPA2 - Software Project Planning KPA1 - Requirements Management
1 Initial	Competent People and Heroics	

Table 2.5. SW-CMM v1.1 Key Process Areas (KPAs) After [Ref. 11]

g. Estimation

Historically, the costs and schedules for most software projects have been greatly underestimated. Many times schedules and costs are dictated by the sponsor, leading to an estimate that is less than adequate. Also, software development efforts are started without a detailed analysis of cost and schedule. And, most sponsors cannot accept the fact that quality software is not cheap. All of these reasons lead to great need for a software estimation process that works and is followed.

The Software Estimation Process is shown in Figure 2.8.

There are several methods that can be used in the estimation process.

- Experience
- Historical Data
- Wideband Delphi Technique
- Pert Sizing
- Function Points
- Automated Sizing Tools

Methods for Wideband Delphi, pert sizing, and function point will be outlined in the database. Automated sizing tools and cost estimating tools, such as SoftEST, COCOMO II, COSTAR, and REVIC will also be available for use.

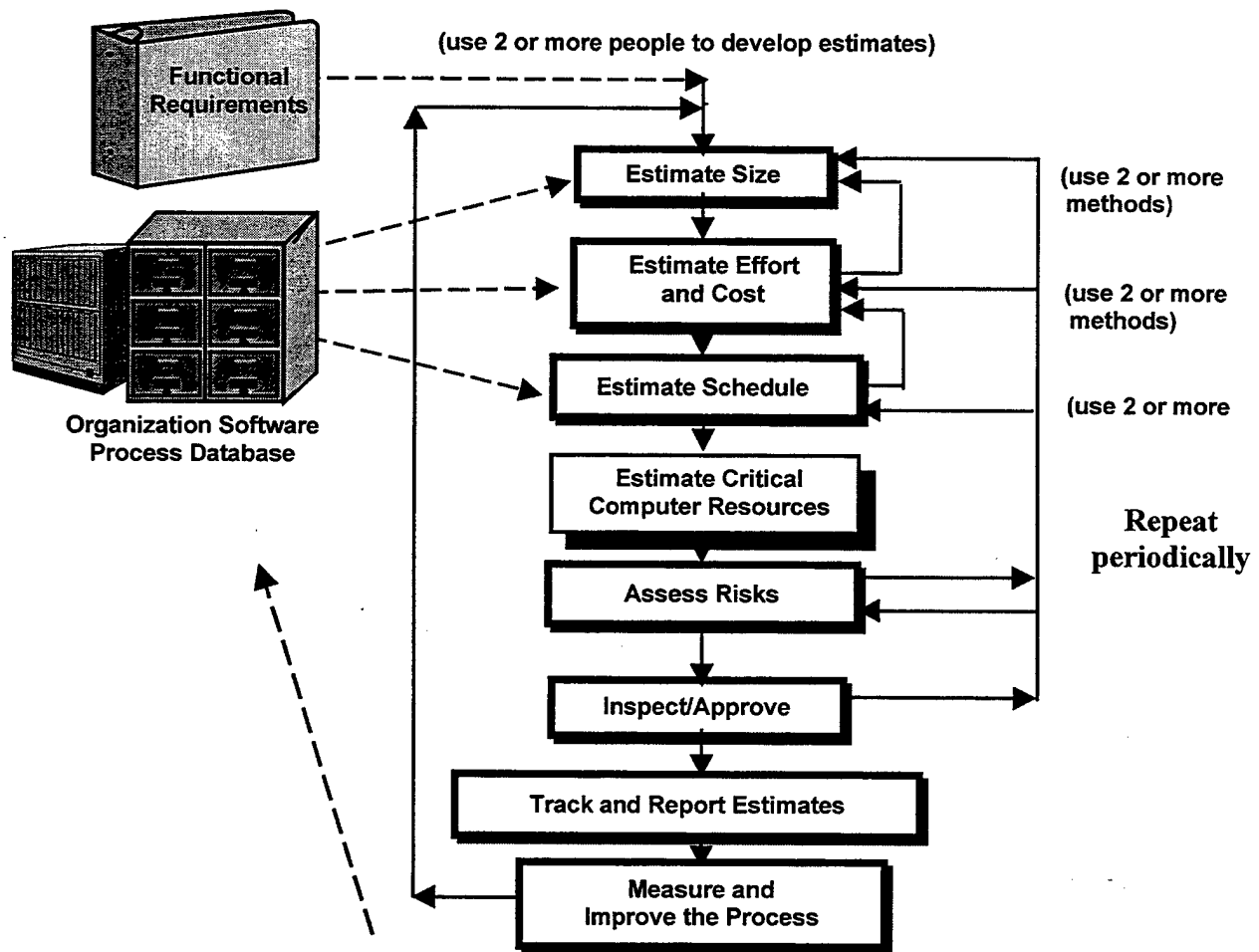


Figure 2.8. Software Estimation Process From [Ref. 21]

Since estimation should be done throughout the life cycle of the development process, estimates should be re-calculated periodically and after major changes requested by the customer. Each updated estimate will be incorporated into the Software Estimation File (SEF). The SEF will include information regarding the estimations, such as, estimation methods used, date of estimate, size, cost, schedule, critical computer resources, and risks for each estimate that is developed. All software estimates are submitted for use in the organization's software process database, as well as any lessons learned for improving the estimation process.

MENTOR will also provide a customizable estimation checklist for all development team levels, which can be used to track the estimation process. This will ensure that process steps are not missed.

h. Lessons Learned

The lessons learned database will contain organizational and project knowledge from a lessons learned standpoint. Common problems, issues, and solutions that have been gathered throughout the organization will be available to the software manager and development team. A troubleshooting guide developed by the SEPO will be available and will include:

- Problem – problem statement
- Reasons – reasons why the problem exists
- Confirmation – ways to confirm a problem
- Solutions – suggested solutions to the problem
- Avoidance – ways to avoid the problem
- Contingencies – suggested contingency plans if the problem has already occurred
- Metrics – suggested metrics to collect and track the problem's consequences.

i. Life Cycle Development Models

There are several life cycle process models the software manager could use to run a software development effort. Depending on the type of project, MENTOR will assist the project manager in selecting a model that will best fit the needs of the customer, development team, application to be developed, time to market, and funding requirements. An overview will be provided for two basic models available for use: (1) Linear Sequential Model or Waterfall and (2) Evolutionary Model.

The Linear Sequential or Waterfall Model is shown in Figure 2.9. This model is more commonly known as the “Waterfall Model.” This model emphasizes a sequential approach to software development that has a clear beginning and end, and appears in varying degrees of phases, but is basically represented by the following diagram.

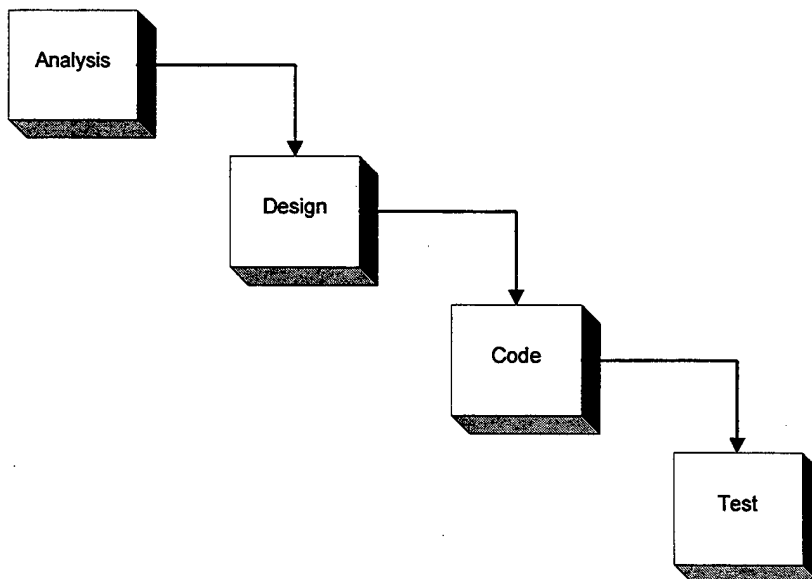


Figure 2.9. Linear Sequential or Waterfall Model After [Ref. 14]

The analysis phase consists of identifying user needs, performing analysis, and defining all requirements. The design phase consists of the actual system and software engineering design. Coding includes implementation of the design phase and some low level testing. The cycle is completed during the test phase where all software modules have been integrated and are then formally tested to meet the initial requirements.

When time to market is the key to success, an evolutionary model may be the answer. This process is iterative in nature and results in a product that can be updated over time but is quick to market. Developers, such as Microsoft, use this development philosophy to catch the wave of technology. If they used a sequential or waterfall model to produce a software product, the need for a particular product may have changed by the time the software was completed and to market. By utilizing an evolutionary development model, they reap the rewards at all stages of product development by releasing updates for each cycle.

There are two basic evolutionary models from which others are further refined.

- Incremental - basically, an iterative waterfall model with each iteration yielding an operational product. This model is used when an early capability is needed, the system allows for natural breaks and the funding and staffing resources are incremental.
- Spiral - originated by Barry Boehm and combines the linear sequential model with an iterative nature. The basic spiral model, as found in ACM SIGSOFT's Software Engineering Notes, is shown in Figure 2.10.

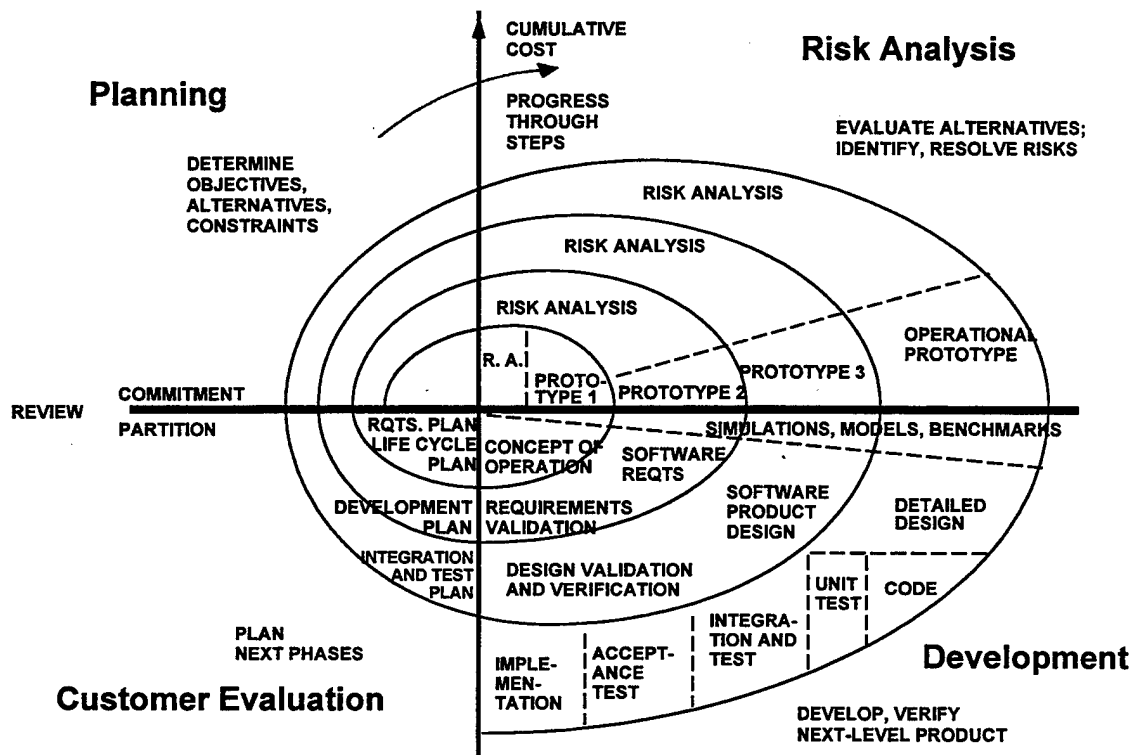


Figure 2.10. Spiral Model From [Ref. 1]

Each process model will be detailed so that MENTOR can guide the manager through the development process every step of the way. Details will include process activities, roles, and responsibilities for following the process model chosen for the development effort. IEEE/EAIA 12207 and other MIL-STDs will also be available as process references for the development team. Also available will be an on-line checklist providing a quick view of items covered and future items to be completed.

j. Tracking and Oversight

Tracking and Oversight provides the visibility needed for the successful completions of a software project. It identifies the methods and tools used for monitoring the project while cycling through the process phases. Each phase also includes weekly or monthly progress meetings, written progress and change reports, and invites the social interactions needed to promote healthy team communication. These methods and tools provide progress assessment and visibility, cost monitoring, and earned value tracking, metrics selection, collection and evaluation.

Metrics provide a quantitative measurement of the process, product and project health, as shown in Figure 2.11. They support risk management, productivity and process improvement, progress tracking, reporting mechanisms and data, and input for future lessons learned. This support helps the project manager and team members by providing the ability to anticipate what can go wrong, support tradeoffs, and evaluation of performance results.

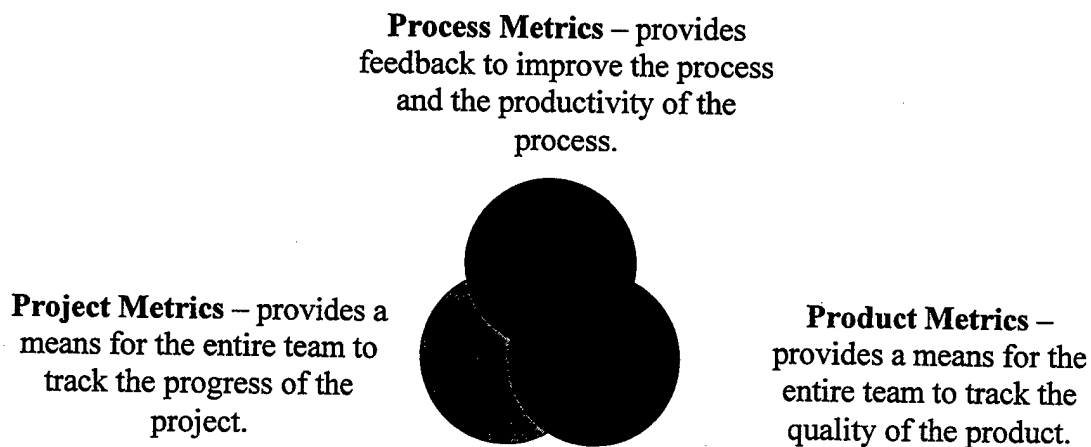


Figure 2.11. Three Areas of Metrics After [Ref. 17]

The SEPO has developed a Software Project Tracking and Oversight (SPTO) Process that MENTOR will use as guidance during the project life cycle. An overview of this process is shown in Figure 2.12.

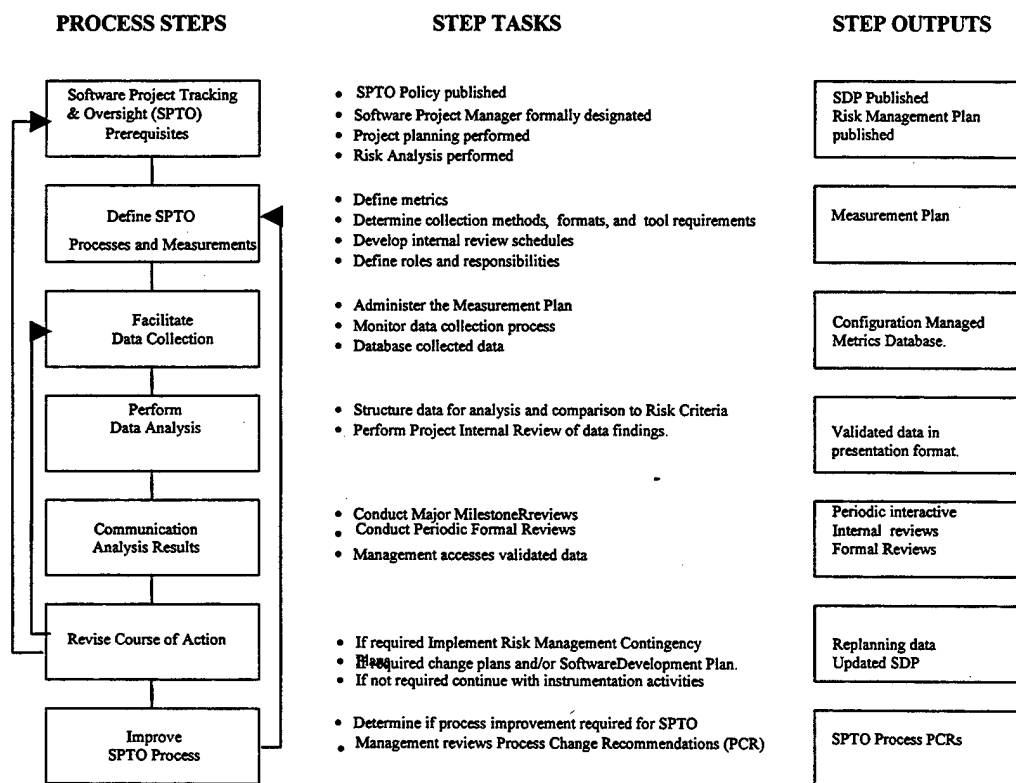


Figure 2.12. SPTO Process Overview From [Ref. 22]

MENTOR will also provide an adaptable SPTO checklist for all development team levels, which can be used to track progress. This will ensure that process steps are not missed. Examples of forms, plans and other miscellaneous documentation will also be provided for reference and tailoring by MENTOR. The list

below contains some of the examples that are currently available for use by MENTOR.

[Ref. 20]

- Software Project Tracking and Oversight Process
- Sample Software Management Plan
- Sample Project Plan
- Sample Monthly Actual Costs Spreadsheet
- Sample Project Tracking Spreadsheet
- Sample Staff Hour Metrics Forms
- Sample Production Engineering Staff Hour Metrics Form
- Requirements Specialist Staff Hour Metrics Form
- Sample SCM Specialist Staff Hour Metrics Form
- Sample Status Data Collection Forms
- Sample Planning Data Collection Forms
- Sample Quarterly Review Requirements
- Earned Value Overview

k. Training

Continuous learning and training is essential for the improvement of a software organization. The training database will contain training course materials, such as briefs, exercises, on-line training guides, interactive courses, and reference material that every level of the software team can use for knowledge growth and process improvement.

Currently, the SEPO has developed much of this material, and it can be found on the SEPO WWW Homepage at sepo.spawar.navy.mil. Interactive guides and tutorials need to be developed based on the Software Program Management Course material and Software Management for Executives Guidebook.

1. Tools

The tools database will contain the tools needed by the user and MENTOR to manage a software development effort. There are many Computer Aided Software Engineering (CASE) tools available that help the user to estimate costs, track requirements, manage software configuration items, monitor project status, and provide the day-to-day office tools needed to support team in communication and development. It is assumed that MENTOR will use the following MS Office Suite of tools:

- Outlook
- Word
- Excel
- Project
- Schedule
- Access

MENTOR will have estimation tools available, similar to the tools listed below, to support the user in estimating cost, effort, and schedule.

- COSTAR - SoftStar Systems
- REVIC v9.2 - Sponsored by the Air Force Cost Analysis Agency
- SoftEST Cost Model v1.1 - Follow on to REVIC
- Cost Xpert v2.1 - The Cost Expert Group

Requirements Management tools will also be available for MENTOR and user needs. The list below is an example of the types of requirements management tools that should be included in the database.

- Requisite Pro - Rational Software
- Dynamic Object-Oriented Requirements System (DOORS) - Quality Systems and Software, Ltd.
- SLATE REquire - TD Technologies
- icConcept-RTM - Integrated Chipware Inc.
- Caliber-RM - Technology Builders, Inc.

MENTOR will also utilize software tools to track and monitor the software development effort. These tools will help the team mitigate, contain, and avoid possible risks by continually monitoring the project. Two examples of the tools available are:

- Risk Radar - Software Program Managers Network
- Project Control Panel - Software Program Managers Network

Configuration Management Tools, such as those listed below, are invaluable to the software development team and will also be available for MENTOR and the team to use.

- ClearCase - Rational Software
- PVCS - MERANT
- RAZOR - Tower Concepts, Inc.

m. Resource Library

The Resource Library database will contain an up-to-date listing of all reference books, and visual and audio media that is available as a resource to the software development team. MENTOR will allow a software development team member to search for reference material availability and provide points of contact and due dates if the reference is checked out. In addition, the resource library will provide a central data repository for all data deliverables that have been approved for incorporation into the OPAD.

2. Project Process Asset Database (PPAD)

The Project Process Asset Database will contain all the artifacts unique to each active project. This includes:

- Tailored Processes
- Collected Metrics
- Engineering Notes and Decision Justifications
- Configuration Controlled Items
 - Requirements, Architectural, Interface, and Design Specifications
 - Management, Development, Project, Quality, Configuration Management, and Test Plans
 - Source Code Modules, System Build and Installation Files
 - Development Procedures
 - Test Procedures and Results
 - User Documentation
 - Data Dictionaries
 - Related Support Tools
 - Logical Data Structures
 - Compilers, Linkers, and Loaders

Once projects are completed, these artifacts are approved for release to the OPAD for resource purposes.

3. Agent Rules and Knowledge Database (ARKD)

The Agent Rules and Knowledge Database contains MENTOR's rule base and algorithms which allow MENTOR's agents to reason, learn, and interact within the system.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MENTOR DESIGN GOALS

Five major design goal areas were chosen for the conceptual design. Software managers throughout SPAWAR Systems Center San Diego provided design goal inputs based on their requirements for an interactive software process management tool. The key areas they expressed interest in were as follows:

- Interactive Project Tutorial
- Project Process Management
- Lessons Learned
- Strategic Planning
- User Interface

A group of SSC software development managers were asked to attend a briefing on MENTOR and then provide input on the capabilities, characteristics, key areas, and type of assistance they would like to see from an intelligent assistant like MENTOR, based on the five areas mentioned above. The MENTOR design goals, in the following sections, are based on those inputs.

A. INTERACTIVE PROJECT TUTORIAL

The managers were asked what key process areas they would like MENTOR to cover in an interactive tutorial for software development. The areas of interest are compiled as follows:

- Software Engineering Process Models and Methods
- Requirements Management
- Design and Engineering
- Testing and Integration
- Inspection Process
- Independent Verification and Validation
- Project Tracking and Oversight
- Estimation Process – Size, Cost, Schedules

- Planning and Schedule Building
- Quality Assurance
- Configuration Management
- Risk Management
- Reviews
 - Project
 - Peer
 - Design Reviews
- Metrics
- Documentation Development
- Contractor Acquisition and Performance Monitoring
- Commercial Off The Shelf (COTS)/Government Off The Shelf (GOTS)
- Capability Maturity Model (CMM)
- Software Engineering Institute (SEI) Certification Guidelines -> Checklist
- Resource Planning and Utilization (the right person for the right job)
- Creating Teams
- People Management
- Team Communications and Meeting Skills
- SPAWAR Center Software Development Policies and Procedures
- Standards
- Continuous Process Improvement Guidelines
- Process Change Management
- Reuse
- Training
- Defect Prevention
- Technology Change Management
- Marketing

In all the areas mentioned above, the software managers wanted “how to” guides, access to templates, information and interactive guides on how to fill out the templates, and samples of existing documentation. They also wanted an extensive knowledge-base that provided a novice the information and guidance necessary on the software development process, from cradle to grave, as well as a quick look tutor and reference assistant for the experienced software manager. The software managers also revealed that they wanted a comprehensive Web based assistant that provided a fun learning

experience and an invaluable resource for every aspect of the software development process.

B. PROJECT PROCESS MANAGEMENT

Several software managers, at SSC, were asked how MENTOR could assist them in their day-to-day management of a software development effort. The managers stated that they wanted a Web-based assistant that could reduce redundancy by sharing all information and tools, promote a teaming environment and open communications, and ensure quality software products by following a repeatable process. MENTOR must have the ability to learn and to improve its methods and the development process for future projects and have the ability to act on behalf of the user, based on authority granted by the user. One manager reflected the need for MENTOR to require justifications as to why a manager chose not to follow a specific process, guidance suggested, or complete a step or request for documentation. This would be valuable input to management metrics and lessons learned.

MENTOR must also have the capability of on-line Help that would use the tutorial interface to provide the information needed. If the information or help requested is not in the tutorial database, MENTOR should take note that this information is required in the tutorial database and seek the information from the system administrator. Also, MENTOR should facilitate 360° feedback mechanisms, promoting open lines of communication between and among all levels.

The software managers stated they wanted MENTOR to guide them through the development process step-by-step, from cradle to grave. MENTOR must know the

development process, steps needed, deliverables and deadlines required to successfully complete the project based on the initial schedule. MENTOR should use the CMM and organizational policies, strategic plans, goals, and objectives for guidance. The guidance offered by MENTOR should be based on the user role and the manager's definition of the team.

Software managers indicated that the following key areas and tasks could be automated and handled by MENTOR, based on user inputs and authority granted:

- Sending e-mails
- Meeting notifications - (should know who needs to attend a meeting then notify)
- Agendas
- Project status reports, stop light charts, earned value
- Distribute status reports if within pre-determined baselines
- Metrics gathering and analysis, data mining
- Notification of deadlines
- Action item lists - creation, request status updates from person assigned the action item, and closure
- Lessons learned information gathering
- Documentation Review, Modification, and Approval Routing
- Prompt for scheduled events

The managers also indicated the need for real-time and on-demand project information. This could be in the form of indicators on the desktop or MENTOR direct interactions. MENTOR should alert the manager to problems (based on trends, baselines, ranges, and schedules), providing project visibility to all users interacting with MENTOR.

C. LESSONS LEARNED

SSC software managers stressed that a good lessons learned knowledge base is essential to the continued success of software development efforts at the center. The

database should be comprehensive, requiring incorporation of all types of data and information regarding the development effort, from every SSC software development project. MENTOR must be able to store, retrieve, classify, organize, search, filter, and extrapolate lessons learned data and information via an easy to use intuitive interface. MENTOR must be able to learn and to continuously improve the software development process based on the lessons learned and justifications, given by the manager, as to why standard processes or tasks were not followed. Managers added that they would like to see features that would provide project troubleshooting capabilities and the ability to suggest review of lessons learned, based on current project status, trends, and decisions being made during the projects life cycle. If information and data requested are not available in the database, MENTOR should query other users in the network for possible inputs, then incorporate this data for future use.

MENTOR should gather lessons learned throughout the development cycle and not just at the end of a project. This will ensure that all lessons learned are incorporated and not forgotten at the end of what are sometimes very long development cycles. Once the project is completed, the lessons learned are then compiled into a report and form the basis for the project's post-mortem.

Based on the five MENTOR design goal areas, managers would like to see the following types of information in the lessons learned:

- Problem descriptions from past and current projects
- Possible solutions and options for each problem description
- Actions taken to resolve the problems and issues
- Past performance data of other projects with similar requirements and deadlines
- Deviations to the troubleshooting effort and improve the troubleshooting guidance

- All deliverables, schedules, estimates, metrics, reports, documentation, engineering notes, and any other by-products of the software development effort - for all SSC software development efforts

D. STRATEGIC PLANNING

Strategic planning is another vital area where managers thought MENTOR could contribute. They wanted MENTOR to guide them through the strategic planning process, beginning with the development of visions, objectives, and goals for the project and team, as well as the organization. MENTOR should be able to identify areas that should be included, offering questions that need to be addressed. MENTOR should allow for “what if” scenarios, based on user constraints of size, cost, schedule, and resources. Based on these constraints, MENTOR should provide options and possible outcomes for each scenario.

The managers revealed that MENTOR should have the capability to evaluate current project status and project possible outcomes if current trends persist. It should provide information regarding the likelihood of project success or failure. This capability should be used for planning and evaluation of total project health by providing the insight needed to find and correct problems and negative trends before they become detrimental to the success of the project. This allows changes to be made that would affect a positive outcome.

The managers also stated that MENTOR should provide a mechanism for resource projection and planning, tradeoff and trend analyses, and cost/size/time estimation. It should provide suggestions and strategies for team building and development, such as, skills and team roles required for project success. MENTOR

should have the capability to suggest actual members based on their availability and time utilization on other projects.

Another key area where managers wanted strategic input was in funds planning. They wanted an automated means to evaluate whether they currently have enough funding to support the entire development effort, the remainder of the development effort, or any specific tasking.

E. USER INTERFACE

MENTOR's user interface is one of the most important design considerations. Managers wanted an interface that is interactive and Web based, unobtrusive on the desktop, intuitive and easy to use providing a "big picture" view of the overall project health, as well as an active communications center for team interactions. They did not want to be overwhelmed with meaningless data, but instead want clear and concise data in understandable snapshot type views. The interface should have a similar look and feel for all user roles, yet configurable to meet the individual user's needs and preferences. It should also provide for on-line help.

The software managers wanted "on-demand" access to all information and data gathered for their project. This included all communications, deliverables, schedules, funding plans, agendas, task and checklists, and any other project information by-products. They also wanted on screen status of their project's overall health by utilizing indicators and alarms that reflect costs, schedule, earned value, and resource utilization, such as stop light charts. Alarms should indicate out of range values that were specified by the user. Along with indicators the user can monitor on the desktop, the managers

wanted MENTOR to interactively supply them with this same information, if an interactive option was selected.

Easy access to all tools needed by the user is another important characteristic of MENTOR's user interface. Not only did managers want easy access; they wanted a common interface for unique tools. In other words, they want MENTOR to interface with a tool so they do not have to learn how to use all possible tools that could aid them in the development effort. Managers also stated the need for a mechanism that supports team communications and aids in the completion of checklists and "to-do" lists based on current and future deadlines.

IV. AGENT BACKGROUND

This chapter seeks to familiarize the reader with the basics for understanding intelligent agents.

A. WHAT IS AN INTELLIGENT AGENT?

There are a variety of descriptions for software agents, each with different characteristics and behaviors. Daniel Weld, in his article *The Role of Intelligent Systems* describes five characteristics that an intelligent agent must possess. [Ref. 25]

- Integrated – Support an understandable consistent interface
- Expressive – Accepts requests in different modes
- Goal-Oriented – Determines how and when to achieve a goal
- Cooperative – Collaborates with the user
- Customized – Adapts to different users

Tecuci's definition of an intelligent agent, as seen below, is a very comprehensive and encompasses the type of agent characteristics that MENTOR will utilize.

A knowledge-based system that perceives its environment (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or other complex environment); reasons to interpret perceptions, draw inferences, solve problems, and determine actions; and acts upon that environment to realize a set of goals or tasks for which it was designed. The agent interacts with a human or some other agents via some kind of agent-communication language and may not blindly obey commands, but may have the ability to modify requests, ask clarification questions, or even refuse to satisfy certain requests. It can accept high-level requests indicating what the user wants and can decide how to satisfy each request with some degree of independence or autonomy, exhibiting goal-directed behavior and dynamically choosing which actions to take, and in what sequence. It can collaborate with its user to improve the accomplishment of his/her tasks or can carry out such tasks on user's behalf, and in so doing employs some knowledge or representation of the user's goals or desires. It can monitor events or procedures for the user, can advise the user on how to perform a task, can train or teach the user, or can help different users collaborate. [Ref. 24]

The remainder of this section will provide a basic foundation for understanding intelligent agents, basic classification, and issues that arise when dealing with distributed agent networks.

B. BASIC AGENT ARCHITECTURE

A basic agent architecture is shown in Figure 4.1. It consists of an environment interface, a reasoning engine, and a knowledge base. The environment interface allows inputs and outputs from the environment, be it a human user, other agents, or application. The reasoning engine carries out the requested tasking through manipulation of data. The knowledge base contains procedures and related data that guide the agent.

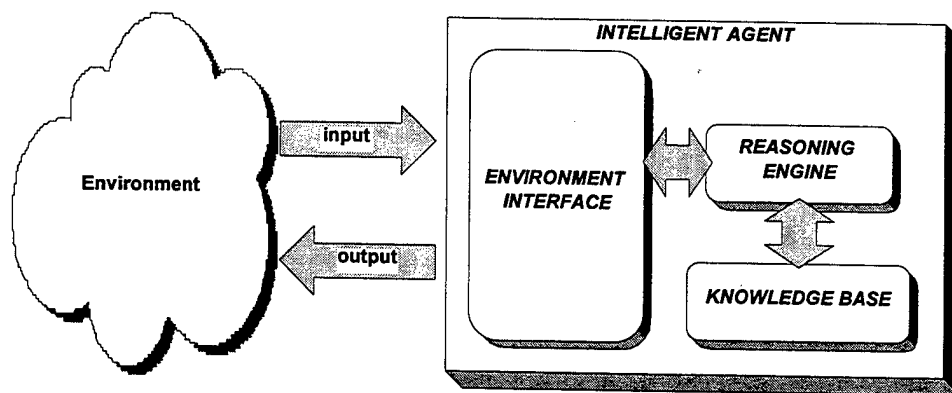


Figure 4.1. Basic Agent Architecture After [Ref. 24]

An agent that independently obtains information is called a learning or adaptive intelligent agent. The learning agent obtains information through its' interactions with the environment. Learning is defined as "the modification of behavior through experience or judgement." [Ref. 24] All tasking performed by the agent is passed from the environment interface to the learning engine where it is processed, while drawing on

the reasoning engine and its resources. Tasks that are learned are then incorporated into the knowledge base for future use and process improvement. A basic learning agent architecture is shown in Figure 4.2.

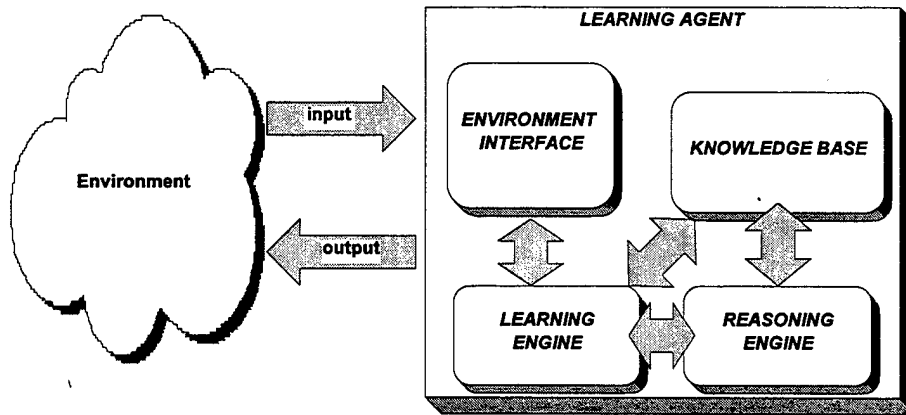


Figure 4.2. Basic Learning Agent Architecture After [Ref 24]

C. AGENT CLASSIFICATION

“A useful classification must have the goal of categorizing existing agent systems and future developments within a standardized scheme.” [Ref. 3] To gain a basic understanding of how agents differ; Bui formulated a classification using eight software agent characteristics, as shown below in Table 4.1.

Software Agent Taxonomy							
Intelligence	Rigid/Automated		Reasoning		Planning		Learning
Mobility	Stationary			Mobile			
Lifetime	Ad hoc		Cloning			Persistent	
Interaction	Agent-to-Agent		Agent-to-Application			Agent-to-User	
Task Specificity	Specific				General		
Behavior	Autonomy	Collaboration	Cooperation	Competitive	Champ	Relay	Crew
Environment	Stable/Secure			Stochastic/Insecure			
Initiative	Push			Pull			

Table 4.1. Spectrum of Software Agent Characteristics After [Ref. 5]

1. Intelligence

The intelligence of a software agent can be described in terms its ability the reason and learn behaviors. [Ref. 2] An agent that is *rigid* is one that performs only simple tasks based on specific instructions from the user and simple rule execution. *Reasoning* allows the agent to make decisions or inferences based on information provided by the user and references from the Organizational Process Asset Database (OPAD). *Planning* agents have the capability to independently plan actions and carry them out to completion. An agent with *learning* abilities has the highest level of intelligence. It provides the ability to learn and adapt to the user and environment, as well as reason. [Ref. 5]

2. Mobility

Mobility is the ability of the agent to navigate through the system. An agent's mobility is either stationary or mobile. Stationary or static agents reside on one computer but can communicate with other agents by sending messages across the network. Mobile agents have the ability to travel from place to place, maintaining all state information, install itself at the remote site, and carry on execution to complete its tasking. [Ref. 5] There are several advantages to using mobile agents, as opposed to stationary agent.

The first advantage is that mobile agents reduce network loading. Mobile agents fulfill their goals by traveling to gather information and perform tasking locally at the remote site, therefore avoiding the usual network message traffic. Another advantage is that since mobile agents act autonomously, a continuous network connection is not required. The network connection is removed once the mobile agents is tasked and sent

across the network. After the agent has fulfilled its goals, it returns and establishes a network connection with the user. In addition, mobile agents have the ability to travel across the network and meet with other agents with similar interests, therefore expanding its capabilities and knowledge. [Ref. 3] Two types of mobile agents exist within the mobility characteristic: mobile scripts and mobile objects. [Ref. 3]

Mobile scripts are agents that are sent to a remote site before the agent executes its tasking. In comparison, mobile objects can move from place-to-place at any time during their task execution, transferring all current state and system environment information along with the agent. [Ref. 3]

3. Lifetime

The temporal nature or lifetime of an agent is the “persistence of identity and state over long periods of time” [Ref. 2] and can be described as ad hoc, cloning, or persistent. An *ad hoc* agent is one that completes a specific task and dies gracefully. A *cloning* agent has the ability to duplicate itself in order to complete a task faster, however; this may cause inter-agent communication and control problems. An agent that is *persistent* does not die after a task has been completed, but instead lives indefinitely. [Ref. 5]

4. Interaction

The interaction capabilities of an agent can be described as agent-agent, agent application, or agent-user. Agents that collaborate with one another work in an *agent-to-agent* relationship. This relationship can be peer-to-peer or hierarchical. Other types of agents communicate with services, databases, utility programs, and any other application. These agents have interactions that are *agent-to-application* specific. Finally, an agent

that interfaces directly with a user to help the user accomplish tasking has *agent-to-user* interactions.

5. Task Specificity

An agent's tasking characteristics are either specific or general. Task *specific* agents are specialized and specifically designed to accomplish one task only. A *general* tasking agent can accomplish many different types of tasking, but is not specialized in any one area and may need to consult other specific tasking agents to complete certain tasks. [Ref. 5]

6. Behavior

An agent's behaviors can be characterized as autonomous, collaborative, cooperative, competitive, champion, relay, or crew type. The first agent behavior, *autonomy*, is "independent, continuously active and not dependent on instructions from its user" [Ref. 3] to complete its tasking. It must have "both control over its actions and internal states and be provided with those resources and capabilities required to perform its tasks." [Ref. 3] The user should specify the level of autonomy based on the specific tasking required. For example, an agent is capable of estimating an increase in cost for a new software requirement and sending a request for notification of funds to a sponsor. However, the user may not want the agent to send the notification without prior approval.

The second behavior is that of collaboration. A *collaborative* agent works with other agents to complete its specified tasking. This type of agent may possibly provide faster and more accurate information based on multi-source inputs and the collaborative

efforts of other agents. [Ref. 5] For example, collaborative agents may meet and decide which cost estimation method is best for the effort at hand.

Cooperation is the third type of behavior. Agents that are cooperative provide the assistance needed for other agents to complete their specified tasking [Ref. 5] and usually concentrate on solving problems. [Ref. 3] An example of this type of interaction is one in which an agent is tasked to check a project's status, then e-mail the status report to several predetermined users. The tasked agent requires the cooperation of an e-mail agent to send the status report to the users. The cooperative effort of "several agents permits faster and better solutions for complex tasks that exceed the capabilities of a single agent." [Ref. 3] All agents in a cooperative effort benefit because each agent's goals are realized in the shortest time possible, either by gaining help from the other agents or having another agent perform the tasking all together. Agents that cooperate must have the capability to communicate their "goals, preferences and knowledge" and therefore require extended communications language capabilities. [Ref. 3]

The fourth and fifth behaviors outlined by Bui are competitive and champion. A *competitive* agent aggressively optimizes itself and is not concerned with other agents achieving their outcomes. [Ref. 5] A *champion* agent is similar, but is at the highest level of importance and cares only about the task outcome and not the method by which it is achieved. Both the competitive and champion agent may consume resources at the expense of other agents.

A *relay* agent, the sixth behavior type, is one that completes its tasking then hands it off, with state information included, to another agent for further processing. [Ref. 5] For example, a relay agent may perform a highly specialized task, such as estimation

using the COCOMO method. After the agent has performed the estimation, the tasking is handed off to another agent that performs a second estimate based on another method.

The final behavior type is crewing. *Crewing* agents work together, simultaneously, to achieve a desired outcome. [Ref. 5] For instance, a manager needs a status report on funding burn rates from every team member. A team of crewing agents would be assembled where each crew agent is responsible for finding the required funding information from a specific team member. Once the information is obtained, the crewing agents report back to the coordinating agent that compiles the information into a report.

7. Environment

An agent's environment is either stable or stochastic. A *stable and secure* environment provides the agent with accurate, predictable, and consistent information in a secure atmosphere. A *stochastic and insecure* environment is a randomly changing environment that is not secure and has a greater chance of inaccurate information being presented. In this case, additional skills may be needed by the agent to overcome information insecurities. [Ref. 5]

8. Initiative

An agent's initiative is either push or pull. An agent that pushes will provide information delivery to the user based on its own discretion. An agent that pulls will provide information delivery to the user at the user's discretion. [Ref. 5]

D. AGENT ISSUES

In order for an intelligent agent to work efficiently within its environment, the following issues must be addressed:

- Language and Platform
- Control
- Resource Management
- Privacy
- Communication
- Mobility
- Understanding

1. Language and Platform

Agents are considered objects having attributes and behaviors that uniquely identify each agent or a class of agents. It is through these attributes and behaviors that other agents will interact, therefore, a language should be used that supports object-oriented development. [Ref. 3] The language used must also allow for graceful halting of execution and provide a means for ease of error handling.

MENTOR agents may be used within various hardware and software configurations, therefore, it is important that agents are developed with platform independence in mind. This is especially important when developing mobile agents that will work in distributed agent environments and travel across networks. [Ref. 3]

2. Control

A control structure must exist that allows the agent to operate freely yet prohibit the possibility for the agent to ignore direction and cause harm to the system. The control

structure must also allow the user to take control of the agent and undo any problems that may have occurred or steps that were taken. [Ref. 2]

3. Resource Management

The agent must be able to manage its resources effectively by using what it needs to complete its tasking, not tie up valuable resources without cause, and share resources with others.

4. Privacy

“Privacy and confidentiality of actions will be among the major issues confronting the use of intelligent agents in our future of a fully interconnected, fully communicating society.” [Ref. 2] It is for this reason that other agents, applications, or users must not be able to compromise the information an agent possesses while it is traveling through the network. However, the owner of the private information may grant access to that private information or give authority for an agent to act on his/her behalf and provide the information.

5. Communication

Agents must have a common, standardized interface that allows effective communication between the user, other agents, and applications.

6. Mobility

The agent design must allow efficient transportation of the agent from machine-to-machine, during execution, with accurate state data. When the agent reaches its

destination, it must pick up where it left off, therefore appearing seamless to the user.

The agent must have the ability to efficiently navigate to all destinations that are possible, possessing authorizations in the form of tickets that allow access to other sites for retrieval of data and transportation of the data back to the requesting site. Mobility has its advantages and disadvantages.

Reduced network load is one significant advantage of using mobile agents. Since the agents travel and execute most of their tasking at the destination point, "data transferred over the network is reduced to a minimum." [Ref. 3] In addition, the agent can filter the data prior to transportation, therefore minimizing the data actually transported. [Ref. 3] Another advantage of a mobile agent is that it operates asynchronously.

Asynchronous operation enables the agent to autonomously complete a task. Once it has been tasked, the agent travels to various sites to solve problems and reach goals. Once the goal is reached, the agent returns with a solution or findings. A network connection is needed only when the agent is transferred across the network. [Ref. 3] Additional advantages can be seen in the decentralized structure that mobile agents support.

Mobile agents not only support the client-server paradigm; they also allow the creation of decentralized structures as well. [Ref. 3] This decentralized structure enables the mobile agent to take alternate paths around bottlenecks or perform tasking on nodes that are less overloaded.

On the other hand, mobile agents have their disadvantages. One major disadvantage is security. The system must ensure the proper identification and

authorization of mobile agents that have access. The system should also provide protection from malicious mobile agents. In addition to security problems there are disadvantages concerning transportation of agents, efficiency, and the use of standard operating environments.

As Brenner, et al states in *Intelligent Software Agents*, complex software and transport layers are needed to move an agent and to perform security identifications and authorizations. This disadvantage may limit the practical use of mobile agents. In addition, efficiency could also be a concern for large numbers of mobile agents, leading to unpredictable network load. Still another disadvantage is that mobile agents do not currently have a standardized system environment or management technique, which is a concern in "the heterogeneous, distributed environments in which mobile agents are normally used." [Ref. 3] These concerns must be weighed against the obvious advantages when designing a system with mobile agents.

7. Understanding

Misunderstandings between an agent and its user should be minimized. There must exist between the user and an agent the ability to understand the interactions that are going on between them. [Ref. 2]

E. MENTOR'S DEVELOPMENT LIFE CYCLE

Building an agent-based system like MENTOR requires the use of a special development model that will provide the necessary steps to define a distributed environment with non-standard and non-communicating system components. Bui describes an architecture that is shown in Figure 4.3.

The first three phases consist of finding the right agents for the tasks needed and the last two phases aim at specifying how the agent will interact in the agent-based system. The first step is to analyze the problem or task. In this phase, requirements and a breakdown of the processes the agent will be involved in must be performed. Step two is to specify the software agent. This involves finding agents or creating agents that satisfy step one. Once steps one and two are completed, the agent profile and behaviors (described using the agent profile in section C above) can be determined, which involves identification of protocols, effective use of resources, and specification and execution of the agent's capabilities for problem solving and data management. Finally, step five determines where the agent fits in to the overall agent-based system.

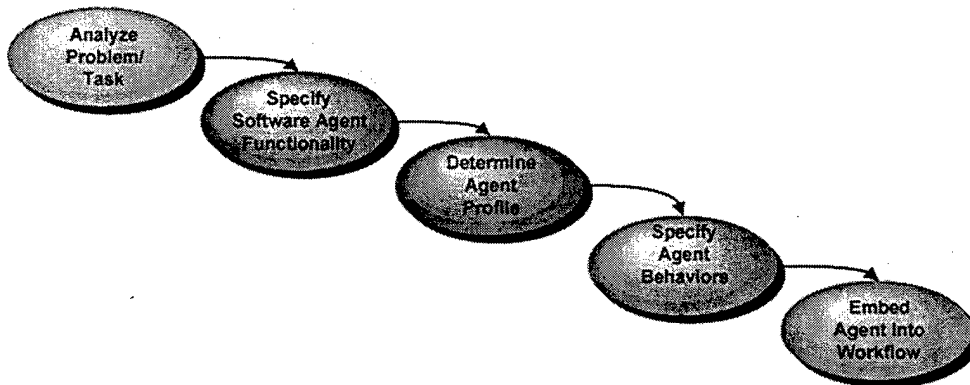


Figure 4.3. A Development Life Cycle for Agent-Based Systems After [Ref. 5]

THIS PAGE INTENTIONALLY LEFT BLANK

V. MENTOR - A SOFTWARE AGENT CONCEPT FOR THE SOFTWARE MANAGEMENT PROCESS

In order to develop a system that will meet our needs, we must first understand how MENTOR will interact with the environment. This will be accomplished through the utilization of a Unified Modeling Language (UML) context diagram. Once we know how the system will interact with its environment, we can specify the functionality performed by system agents with UML use case diagrams. Agent profiles and behaviors will be determined using Bui's agent profile taxonomy from section C, Chapter III, of this thesis. The final step will be to embed the agents in a conceptual MENTOR agent architecture.

A. SYSTEM CONTEXT

The system context is "a map of the world of interest to the system." [Ref. 99] In UML the context diagram represents this view. It shows the system surrounded by the real world objects that interact with the system. The MENTOR context diagram is shown in Figure 5.1.

The real world objects that interact with MENTOR will be the software project manager, development team, system administrator, and the Internet. The manager will interact with MENTOR to request information and data, perform searches, set user preferences, receive information and data, and check program status alerts and alarms. The development team depicted in the diagram represents multiple single users, all with relatively similar system needs. They will request and receive information and data, observe project alerts and alarms, review project status, perform searches, and set user

preferences. The Internet is an object that interacts with MENTOR through the use of a mobile agent performing request and retrieval of specific information and data. The system administrator is responsible for setting up the MENTOR resource databases and agent architecture. The system administrator requires access to all the capabilities of the other objects.

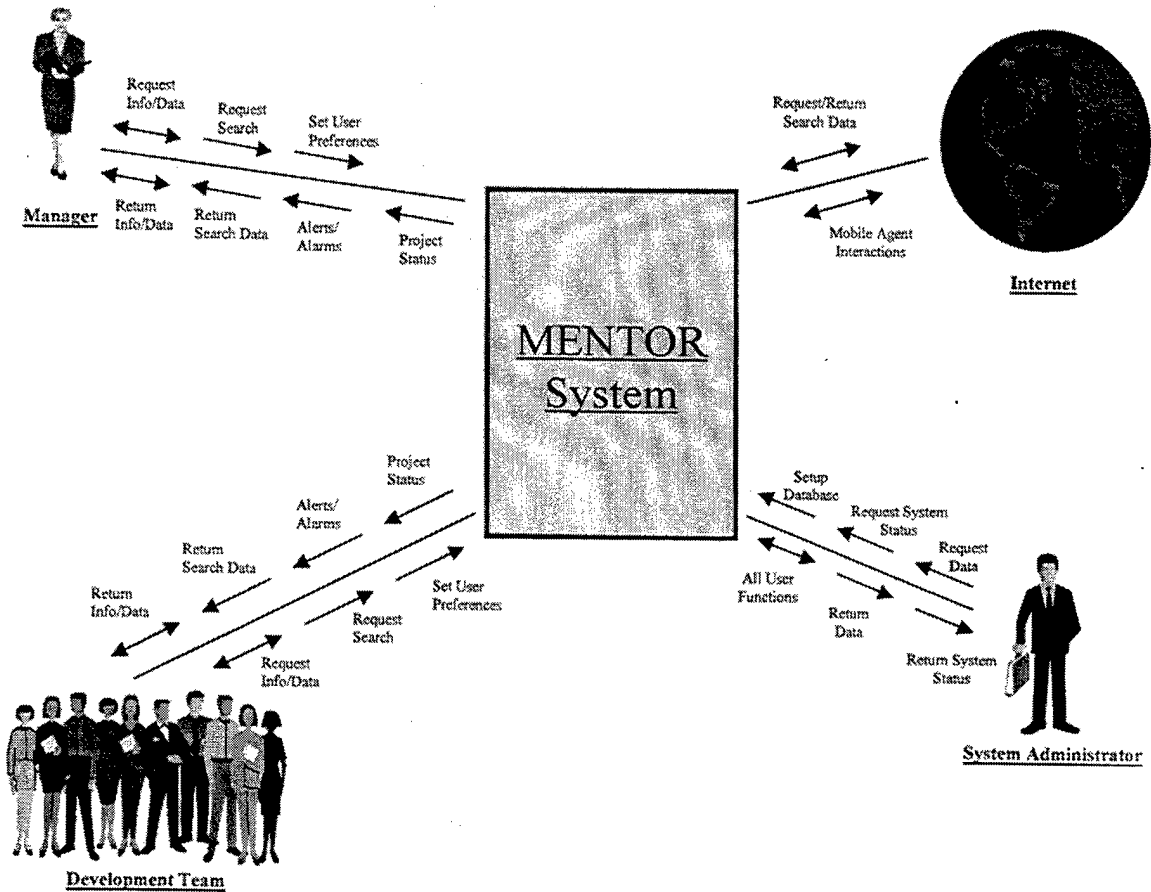


Figure 5.1. MENTOR Context Diagram

B. AGENT FUNCTIONALITY

The MENTOR system use case diagram is shown in Figure 5.2. The use case diagram “shows the general cases of interaction among the system and the external objects.” It provides a big picture view of the main system functions and will enable us to specify top-level agent functionality from step two of the agent development life cycle.

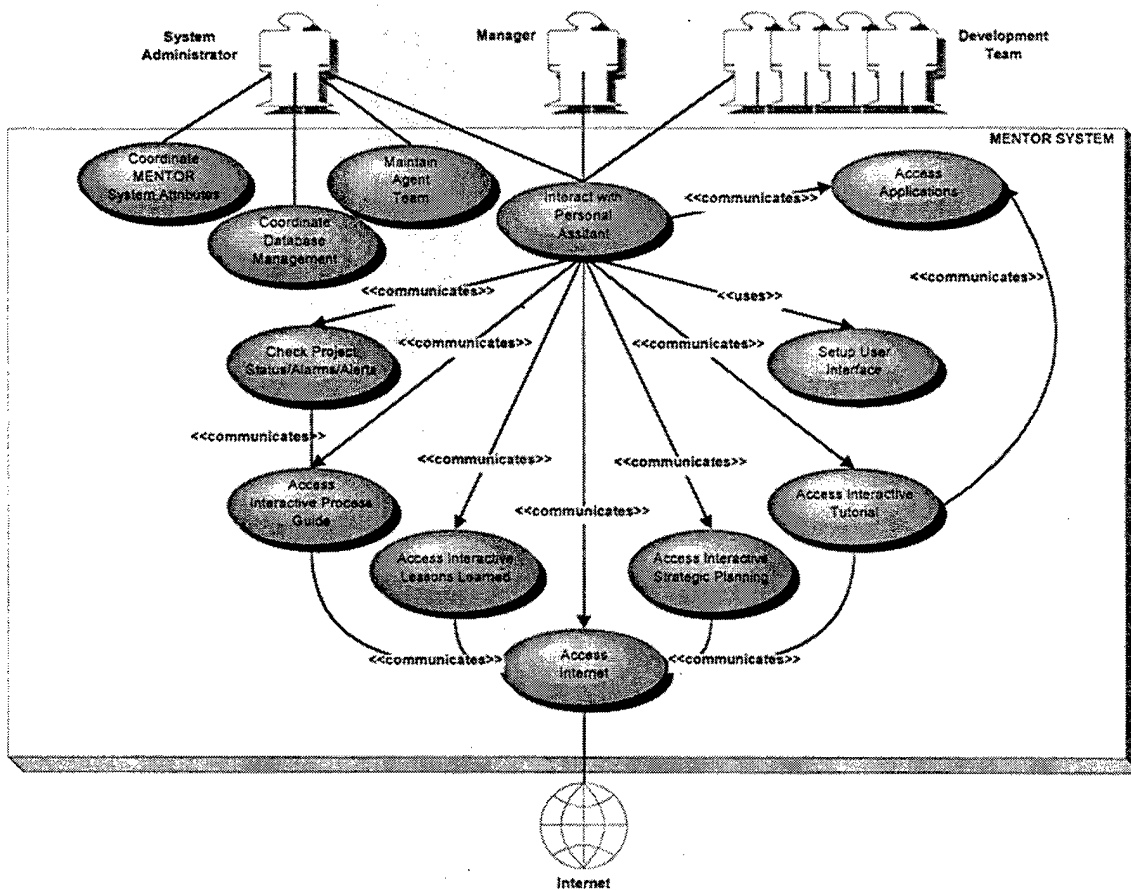


Figure 5.2. MENTOR Use Case Diagram

The top-level functions for a manager or a member of the development team are very similar. The system administrator will have access to all manager and development

team functions, as well as system management functions. The use cases used in the use case diagram and descriptions of each are provided below in Table 5.1.

MENTOR Top Level Use Cases		
UC	Use Case Name	Use Case Description
1	Interact with Personal Assistant	Provides all user interface capabilities and access to all other system functions
2	Setup User Interface	Allows configuration, through the personal assistant, of all user interfaces including display modes and status/alert/alarm setup
3	Check Project Status/Alarms/Alerts	Allows monitoring of project status alarms, and alerts, which the user has setup via the personal assistant
4	Access Interactive Process Guide	Allows interactions between MENTOR and the user, via the personal assistant, to effectively setup and manage a project to successful completion
5	Access Interactive Strategic Planning	Allows MENTOR, via the personal assistant, to provide help, planning, estimation, tradeoff and trend analysis based on constraints entered or current project status
6	Access Interactive Lessons Learned	Allows the user, via the personal assistant, to add lessons learned, or request information on previous lessons learned from all projects supported by MENTOR
7	Access Interactive Tutorial	Allows mentor to provide tutorials and process information via the personal assistant
8	Access Internet	Provides access, through the personal assistant, to and from the Internet for requesting, sending, and retrieving information
9	Access Applications	Provides access, through the personal assistant, to all system accessible applications (MS Word, PowerPoint, and Excel)
10	Maintain Agent Team	Allows functions for direct creation, modification, deletion, and tracking of system agents by the system administrator
11	Coordinate Database Management	Allows direct setup and maintenance function for the system databases by the system administrator
12	Coordinate MENTOR System Attributes	Allows direct modification of user interface and all other system functionality by the system administrator

Table 5.1. Mentor Top Level Use Cases

C. AGENT PROFILES AND BEHAVIORS

The Bui taxonomy described in Section C, Chapter IV, is used in this section to outline agent profiles and behaviors. Agent profiles are provided for all the agents needed to fulfill the functionality of the top-level use case diagram given in the previous section. In addition, lower level agents will be profiled that are needed to provide more complete system functionality. For the purpose of this conceptual approach, the environment is assumed to be stable and secure for all agents involved, operating in a controlled network structure. The following sections are organized by use case, as referenced in the previous section. Additional sections are then provided to profile the remaining agents.

1. Use Case 1 – Interact with Personal Assistant

The User Personal Assistant (UPA) agent is the only agent that interfaces with the user directly and therefore has agent-to-user interactions. Interactions between the agent and user take place via a visual interface that utilizes a Web browser environment that acts as a communication command center for direct team interactions. All subordinate agents must communicate with this agent to fulfill the user's requests through agent-to-agent interactions and collaboration. The UPA agent provides access to all system functions and graphical display of project status, alerts, and alarms. Requests for information, display of requested information, and output capabilities are also provided via the interface as well. This agent functions at the highest level requiring the ability to learn from the user and system interactions. It is stationary because it resides on the user's computer and is persistent because it must constantly monitor interactions between

the user and the system. The UPA agent must have a broad functionality and therefore has general task specificity. Both push and pull initiatives are used, since the agent must provide information when it desires, as well as upon user request. The User Personal Assistant agent profile is given in Table 5.2.

User Personal Assistant							
Intelligence	Rigid/Automated		Reasoning		Planning		Learning
Mobility	Stationary			Mobile			
Lifetime	Ad hoc		Cloning			Persistent	
Interaction	Agent-to-Agent		Agent-to-Application			Agent-to-User	
Task Specificity	Specific				General		
Behavior	Autonomy	Collaboration	Cooperation	Competitive	Champ	Relay	Crew
Environment	Stable/Secure			Stochastic/Insecure			
Initiative	Push			Pull			

Table 5.2. User Personal Assistant Agent Profile.

2. Use Case 2 – Setup User Interface

The Setup User Interface (SUI) agent is used by the User Personal Assistant to provide expert assistance on setting up the user interface. The user can specify how the information will be displayed, what information will be displayed, and when it will be displayed. Ranges for project alerts and alarms, as well as user preferences for color and layout are also specified using this agent. The SUI agent is rigid or automated in its functions because it performs specific tasks that do not require reasoning capabilities. It is a mobile script type agent with the ability to package itself up and travel to the user's computer for execution to setup the interface. Once the agent has fulfilled its tasking it dies gracefully and therefore has a lifetime that is ad hoc. If at a later time the user needs to change the setup configuration, the UPA agent creates another SUI agent to perform the same functions using agent-to-agent interactions. This agent has autonomy, acting

alone to setup the user interface in a push/pull fashion. The Setup User Interface agent profile is given in Table 5.3.

Setup User Interface Agent							
Intelligence	Reasoning		Planning		Learning		
Mobility	Stationary			Mobile			
Lifetime	Cloning			Persistent			
Interaction	Agent-to-Application			Agent-to-User			
Task Specificity	General			General			
Behavior	Collaboration	Cooperation	Competitive	Champ	Relay	Crew	
Environment	Stochastic/Insecure			Stochastic/Insecure			
Initiative	Push			Pull			

Table 5.3. Setup User Interface Agent Profile

3. Use Case 3 – Check Project Status/Alarms/Alerts

The Check Project Status/Alarms/Alerts (CPS) agent is the expert at monitoring user specified status objectives, alarms, and alerts based on guidelines and ranges set during user interface setup. It communicates with system agents in agent-to-agent interactions and has the ability to gather information regarding the requested status, alerts and alarms and compare them, using reasoning, with those set by the user for proper user interface notification via the UPA agent. This agent is a task specific mobile object that travels throughout the network to find the information needed and is persistent due to the need for constant project monitoring. This agent must have the ability to communicate and collaborate with other agents that will provide information. In addition, the CPS agent operates with push/pull initiatives with other agents, providing information to the UPA agent when requested as status and pushing information to the UPA agent when alerts or alarms arise. The Check Project Status/Alarms/Alerts agent profile is given in Table 5.4.

Check Project Status/Alarms/Alerts Agent							
Intelligence	Rigid/Automated	Reasoning		Planning		Learning	
Mobility	Stationary			Mobile			
Lifetime	Ad hoc	Cloning			Persistent		
Interaction	Agent-to-Agent		Agent-to-Application		Agent-to-User		
Task Specificity	Specific			General			
Behavior	Autonomy	Discretion	Cooperation	Competitive	Champ	Relay	Crew
Environment	Stable/Secure			Stochastic/Insecure			
Initiative	Push			Pull			

Table 5.4. Check Project Status/Alarms/Alerts Agent Profile

4. Use Case 4 – Access Interactive Process Guide

The Interactive Process Guide Coordinator (IPGC) agent coordinates all interactions that guide the user through the development process. It must have the ability to reason, plan, and learn from agent-to-agent interactions to guide the user and improve the process. This agent should be implemented as a mobile object to reduce network loading by going to the source of information rather than sending messages across the network and tying up resources. The IPGC agent should have a lifetime that is both cloning and ad hoc. Cloning will enable the IPGC agent to service multiple UPA agents. A lifetime that is ad hoc allows the UPA agent to create an IPGC agent based on the user's needs for process guidance. Before the agent dies gracefully, it must update the IPGC agent's central knowledge base for future assistance. The IPGC agent has the ability to collaborate with other agents to achieve its desired outcome. It has both a push and pull information delivery system that provides information via the UPA agent based on user requests, as well as pushing information to the UPA agent that it deems necessary to successfully complete the project. In addition, the IPGC agent is general in its overall knowledge and task specificity, but has the capability to create task specific mobile

agents to perform specific functions. The Interactive Process Guide Coordinator agent profile is given in Table 5.5.

Interactive Process Guide Coordinator							
Intelligence	Rigid/Automated	Reasoning		Planning		Learning	
Mobility	Stationary						Mobile
Lifetime	Finite			Persistent			
Interaction	Agent-to-Agent			Agent-to-Application		Agent-to-User	
Task Specificity	General						General
Behavior	Autonomy	Cooperation	Cooperation	Competitive	Champ	Relay	Crew
Environment	Stochastic/Insecure			Stochastic/Insecure			
Initiative	Push			Pull			

Table 5.5. Interactive Process Guide Coordinator Agent Profile

5. Use Case 5 – Access Interactive Strategic Planning

The Interactive Strategic Planning Coordinator (ISPC) agent coordinates all activities involving strategic planning. It must have the ability to reason, plan, and learn so that it can provide projections for “what-if” scenarios, improve the strategic planning process, and perform better planning in the future. This agent is a mobile object created by the UPA agent in an ad hoc manner when the user requires assistance. This mobility allows the ISPC agent to gather all current project status information, at the information source, for use in its projections and minimize network loading due to messages. This agent also provides information delivery based on user requests by collaborating with other agents. In addition, it has the ability to create mobile agents to fulfill specific tasking needs and clone itself to support the needs of multiple UPA agents. The Interactive Strategic Planning Coordinator agent profile is given in Table 5.6 below.

Interactive Strategic Planning Coordinator							
Intelligence	Rigid/Automated	Reasoning	Planning	Learning			
Mobility	Stationary						Mobile
Lifetime		Learning					Persistent
Interaction		Agent-to-Application					Agent-to-User
Task Specificity							General
Behavior	Autonomy	Collaboration	Cooperation	Competitive	Champ	Relay	Crew
Environment							Stochastic/Insecure
Initiative	Push						Pull

Table 5.6. Interactive Strategic Planning Coordinator Agent Profile

6. Use Case 6 – Access Interactive Lessons Learned

The Interactive Lessons Learned Coordinator (ILLC) agent coordinates all interactions regarding lessons learned. It must have the ability to reason and learn, in order to provide process improvement for the lessons learned process. The ILLC agent must collaborate with other agents to gather and incorporate lessons learned data into the OPAD. This agent is ad hoc and initiated simultaneously with the IPGC and ISPC agents to constantly monitor the process and strategic planning operations for lessons learned. The ILLC agent dies gracefully when the IPGC and/or ISPC agents are no longer needed. The UPA agent can also initiate the ILLC agent independently to provide historical lessons learned information based on user requests or direct input of new lessons learned data. This agent should be developed as a mobile object to minimize network loading by traveling to the IPGC or ISPC agent to execute. As in the IPGC and ISPC agent profiles, the ILLC agent has the capability to clone itself to support multiple UPA agents and to create task specific mobile agents that realize specialized functions. In addition, this agent must provide information using both push and pull delivery, allowing for user

requests and automatic information gathering. The Interactive Lessons Learned Coordinator agent profile is given in Table 5.7.

Interactive Lessons Learned Coordinator							
Intelligence	Rigid/Automated	Recomm	Planning	Learning			
Mobility	Stationary			Mobile			
Lifetime		Online		Persistent			
Interaction	Human-Agent	Agent-to-Application		Agent-to-User			
Task Specificity				General			
Behavior	Autonomy	Collaborative	Cooperation	Competitive	Champ	Relay	Crew
Environment	Subsistence					Stochastic/Insecure	
Initiative	Push						Pull

Table 5.7. Interactive Lessons Learned Coordinator Agent Profile

7. Use Case 7 – Access Interactive Tutorial

The Interactive Tutorial Coordinator (ITC) agent coordinates all lesson plans and delivery of requested templates and samples. It must have the ability to learn from the environment by recognizing information requested by the user, that is not currently in the database. This mobile object should seek information sources for the unknown data from other users in the network and sources on the Internet. The information should then be provided to the requesting user and incorporated into the database for future user needs. This agent is ad hoc providing information when it is created and accessed by the UPA agent based on need. It must also have the ability to collaborate with other agents to achieve tutorial outcomes and create task specific mobile agents to aid in specialized information requests. This agent's information delivery system is mainly pull, but can also be push to provide data to the user during tutorials when it thinks the users should see it. In addition, the ITC agent should have the ability to clone itself in order to tutor

multiple users through individual UPA agents. The Interactive Tutorial Coordinator agent profile is given in Table 5.8.

Interactive Tutorial Coordinator							
Intelligence	Rigid/Automated		Reasoning		Planning		Learning
Mobility	Stationary			Mobile			
Lifetime	Ad-hoc		Cloning			Persistent	
Interaction	Agent-to-Agent		Agent-to-Application			Agent-to-User	
Task Specificity	Specific			General			
Behavior	Autonomy	Collaboration	Cooperation	Competitive	Champ	Relay	Crew
Environment	Stable/Secure			Stochastic/Insecure			
Initiative	Push			Pull			

Table 5.8. Interactive Tutorial Coordinator Agent Profile

8. Use Case 8 – Access Internet

The Access Internet Facilitator (AIF) agent is responsible for facilitating information flow to and from the Internet. This agent is persistent because it must maintain information flow integrity out of a closed system. It assists other agents in using the Internet, sending and receiving data, searching for information, and ensuring that access is denied to unauthorized mobile agents and outside requests for information. The Access Internet Facilitator agent profile is given in Table 5.9.

Access Internet Facilitator								
Intelligence	Rigid/Automated		Reasoning		Planning		Learning	
Mobility	Stationary		Mobile					
Lifetime	Ad hoc		Cloning			Persistent		
Interaction	Agent-to-Agent		Agent-to-Application			Agent-to-User		
Task Specificity	Specific			General				
Behavior	Autonomy	Collaboration	Cooperation	Competitive	Champ	Relay	Crew	
Environment	Stable/Secure			Stochastic/Insecure				
Initiative	Push			Pull				

Table 5.9. Access Internet Facilitator Agent Profile

9. Use Case 9 – Access Applications

The Specific Applications (SA) agent provides an interface to system applications, such as MS Word, PowerPoint, Excel, and Access. It also provides a common interface for estimation, configuration management, risk management, and other unique CASE tools. This agent must have the ability to provide an automated environment with reasoning and learning capabilities. This will allow the SA agent to determine if the information being entered is correct and sufficient, as well as the ability to learning through interactions, how to improve the interface. In addition, this agent provides the user with a common look and feel for unique tools. The SA agent is also mobile.

Mobility is realized through an agent implementation that is in the form of a mobile object that can travel to where the application resides, perform its duties, and return with information. The UPA agent creates an ad hoc SA agent, with the ability to clone itself to service multiple UPA agents. Also, the SA agent interacts with other agents, as well as applications to reach its end goals and is task specific, meaning each specific agent knows how to interact with a specific application. In addition, this agent has behaviors that are autonomous, collaborative, and cooperative, and are determined based on how the user wishes to use the intended application. Furthermore, information delivery is both push and pull depending on the interactions required. The Specific Application agent profile is given in Table 5.10.

Specific Application Agent							
Intelligence	Reasoning	Planning	Learning				
Mobility	Stationary					Mobile	
Lifetime					Persistent		
Interaction					Agent-to-User		
Task Specificity					General		
Behavior	Autonomous	Collaborative	Cooperative	Competitive	Champ	Relay	Crew
Environment					Stochastic/Insecure		
Initiative					Push		

Table 5.10. Specific Application Agent Profile

10. Use Case 10, 11, and 12 – Maintain Agent Team, Coordinate Database Management, and Coordinate MENTOR System Attributes

The System Maintenance (SM) agent allows the system administrator, through the UPA agent, to maintain the MENTOR agent team, to manage the database, and to modify system attributes. This agent has the ability to reason in order to determine if the system is setup and running properly and to learn from the system administrator's actions how it can provide the best possible assistance. The SM agent must be mobile so that it can travel throughout the system to gather, manage, and maintain the agents and databases based on administrator inputs. These interactions require that the SM agent have the ability to interact not only with the user, but also with other agents and applications. In addition, this agent must be persistent so that it can monitor the system for problems and alert the administrator if assistance is required. Due to the nature of this agent's goals it must have the ability to act autonomously, collaboratively, cooperatively, and competitively based on the system administrator's direction and requests. Likewise, it provides information in both push and pull delivery. The System Maintenance agent profile is given in Table 5.11.

System Maintenance Agent								
Intelligence	Rigid/Automated	Reasoning		Planning		Learning		
Mobility	Stationary			Mobile				
Lifetime	Ad hoc	Cloning			Persistent			
Interaction	Agent-to-Agent		Agent-to-Application			Agent-to-User		
Task Specificity	Specific				General			
Behavior	Autonomy		Collaboration	Competition	Competitive	Champ	Relay	Crew
Environment	Stable/Secure				Stochastic/Insecure			
Initiative	Push			Pull				

Table 5.11. System Maintenance Agent Profile

11. Project Process Asset Database Facilitator Agent

The PPAD Facilitator (PF) agent is the facilitator for all project specific assets, retrieving and delivering information to the appropriate project database, as well as directing mobile agents to the desired project database. This agent is rigid and acts cooperatively as the traffic coordinator for the mobile agents and message traffic that require access to the PPAD's information. It has specific knowledge of the databases and information in the PPAD and should be implemented as a persistent mobile object that can traverse the individual project asset databases to update its view of the information it oversees. The PPAD Facilitator agent profile is given in Table 5.12.

PPAD Facilitator								
Intelligence	Rigid/Automated		Reasoning		Planning		Learning	
Mobility	Stationary							Mobile
Lifetime	Ad hoc		Cloning			Persistent		
Interaction	Agent-to-Agent		Agent-to-Application			Agent-to-User		
Task Specificity	Specific				General			
Behavior	Autonomy	Collaboration	Competition		Competitive	Champ	Relay	Crew
Environment	Stable/Secure				Stochastic/Insecure			
Initiative	Push			Pull				

Table 5.12. PPAD Facilitator Agent Profile

12. Organizational Process Asset Database Facilitator Agent

The OPAD Facilitator (OF) agent facilitates information retrieval and delivery to the organizational databases in the same manner as the PF agent. It cooperatively directs and assists mobile agents and message traffic enabling efficient project asset database access. This agent maintains specific knowledge of its information databases and should be implemented as a persistent mobile object that can traverse the OPAD to update its view of the information it oversees. The OPAD Facilitator agent profile is given in Table 5.13.

OPAD Facilitator							
Intelligence	Reasoning		Planning		Learning		
Mobility	Stationary		Mobile				
Lifetime	Ad hoc		Cloning		Persistent		
Interaction	Agent-to-Agent		Agent-to-Application		Agent-to-User		
Task Specificity	Specific		General				
Behavior	Autonomy	Collaboration	Cooperation	Competitive	Champ	Relay	Crew
Environment	Stochastic/Insecure		Stochastic/Insecure				
Initiative	Push		Pull				

Table 5.13. OPAD Facilitator Agent Profile

13. Agent Management Coordinator Agent

The Agent Management Coordinator (AMC) agent is a persistent mobile agent that monitors all the agents in the network. It performs planning, maintenance, and management tasks regarding agent lifetime, execution, and performance. In addition, it can search for lost agents in the system, and therefore, requires a mobile object implementation. This agent is essential in the management of task execution. [Ref. 106]

The Agent Management Coordinator agent profile is given in Table 5.14.

Agent Management Coordinator								
Intelligence	Reasoning		Planning		Learning			
Mobility	Stationary		Mobile					
Lifetime	Ad hoc		Cloning		Persistent			
Interaction	Agent-to-Application			Agent-to-User				
Task Specificity				General				
Behavior	Collaboration		Cooperation	Competitive	Champ	Relay	Crew	
Environment				Stochastic/Insecure				
Initiative				Pull				

Table 5.14. Agent Management Coordinator Profile

14. Database Agents

The Database Agents listed below all have the same attributes and behave in a similar manner. They are rigid database agents that have specific knowledge of their own database and its contents. They retrieve or deliver information directly into the database in a specified format. They are stationary but persistent agents that cooperate with the OF and PF agents to provide information delivery services to requesting mobile agents or fulfill information requests and deliveries via message traffic. The following list of agents is profiled in Table 5.15.

- Project Planning Agent - PPA
- Requirements Definition Agent - RDA
- Risk Management Agent - RMA
- Configuration Management Agent - CMA
- Quality Management Agent - QMA
- CMM Agent - CA
- Estimation Agent - EA
- Lessons Learned Agent - LLA
- Life Cycle Development Agent - LCDA
- Oversight and Tracking Agent - OTA
- Training Agent - TA
- Tools Resource Agent - TRA
- Resources Agent - RA
- Project Database Agent - PDA

Database Agent										
Intelligence	Rigid/Automated			Reasoning		Planning		Learning		
Mobility	Stationary					Mobile				
Lifetime	Ad hoc			Cloning			Persistent			
Interaction	Agent-to-Agent			Agent-to-Application				Agent-to-User		
Task Specificity	Specific					General				
Behavior	Autonomy		Collaboration		Cooperation		Competitive	Champ	Relay	Crew
Environment	Stable/Secure					Stochastic/Insecure				
Initiative	Push					Pull				

Table 5.15. Database Agent Profile

15. Task Specific Mobile Agents

Task Specific Mobile (TSM) agents are created by other agents to provide specific services not already available from existing agents. They are rigid, mobile, and ad hoc, existing only long enough to fulfill their tasking. These agents have behaviors ranging from autonomy to cooperation depending on the type of tasking. In addition, they can operate in either a push or pull configuration depending on the services it provides. The Task Specific Mobile agent profile is given in Table 5.16.

Task Specific Mobile Agent										
Intelligence	Reasoning			Planning			Learning			
Mobility	Stationary				Mobile					
Lifetime	Cloning			Persistent						
Interaction	Agent-to-Application				Agent-to-User					
Task Specificity	Specific				General					
Behavior	Autonomy	Collaboration	Cooperation	Competitive	Champ	Relay	Crew			
Environment	Stable/Secure				Stochastic/Insecure					
Initiative	Push				Pull					

Table 5.16. Task Specific Mobile Agents

D. CONCEPTUAL MENTOR AGENT ARCHITECTURE

The agent architecture proposed in this section is a result of the final step in the development life cycle for agent-based systems and is shown in Figure 5.3. The architecture is distributed providing many autonomous expert agents that act together and appear to the user as one agent. The purpose of using an array of expert agents, instead of one or two all-encompassing agents, is that it would require a large amount of overhead and resources to run one large program that would also be very slow.

Agents work together by advertising themselves and their capabilities to the other agents in the system, periodically. Each time a new agent is created or spawned, it must immediately advertise its capabilities, services, location, and interface requirements across the network. This allows the Agent Management Coordinator to track the active agents and encourage efficient utilization of resources and inform the other agents of new capabilities. If the Coordinator has not heard from an agent that was active, it will search for the agent, determine if the agent is still needed and continue monitoring, or end the life cycle of the agent (freeing resources). If an agent requires a service that it does not have knowledge of, the agent can ask the Agent Management Coordinator for assistance in locating agents that provide the desired tasking.

Communications between agents is accomplished through request and reply messages. A request contains specific identification information, such as request ID, contact information, rules and format for interactions, and the requesting agent's ID. A reply must contain the reply agent's ID and the request ID. This provides a means of distinguishing replies to multiple simultaneous requests. [Ref. 8]

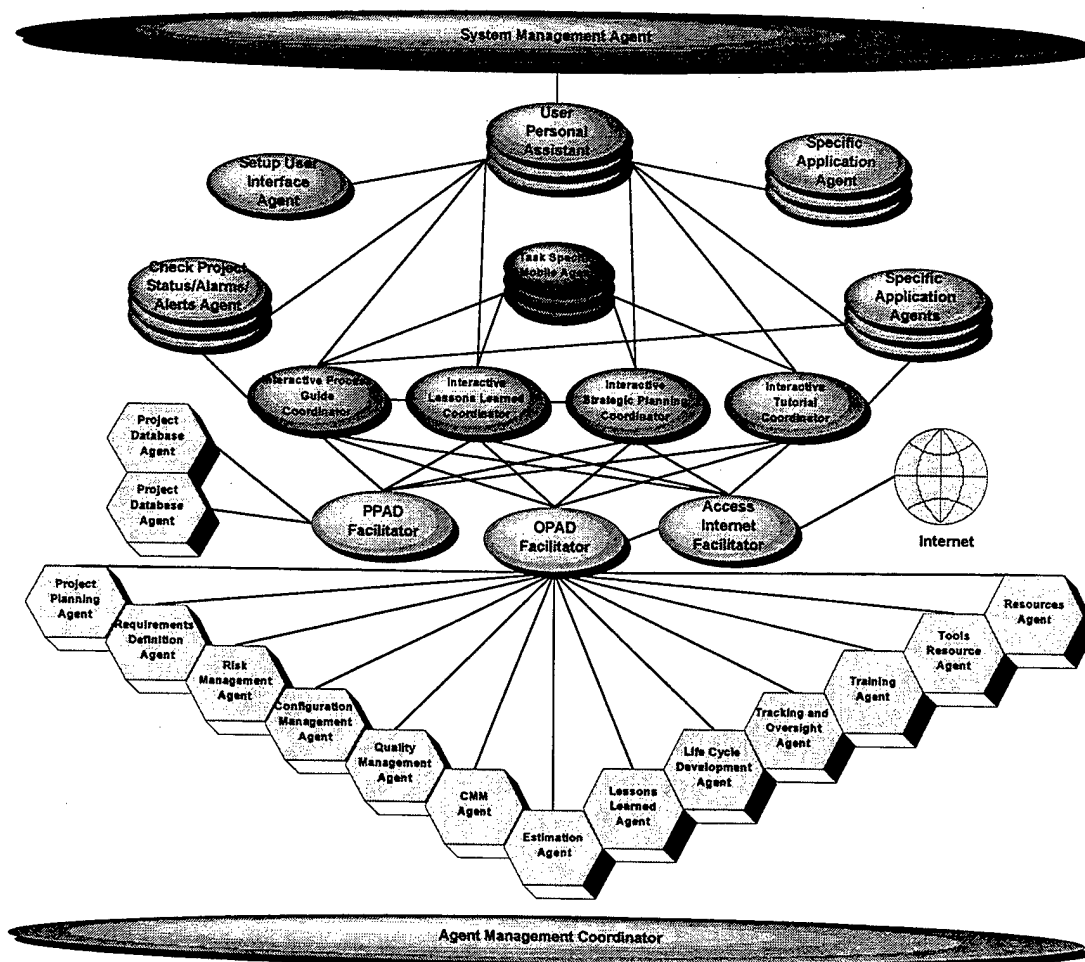


Figure 5.3. MENTOR Conceptual Agent Architecture

The conceptual agent architecture is a team of agents working with several lead coordinators and one user assistant for each user. The User Personal Assistant is a Web-based browser interface that allows information flow between the user and MENTOR. The user may request information, provide information, and visually track the progress of the project via the interface. The User Personal Assistant tasks other agents to gather information, format information, report information, store information, and perform tasks based on user requests and project tracking requirements. In addition, the User Personal

Assistant should be programmed to accept natural language interactions, allowing the user to easily interact with the system without having to learn a specific rule set.

The User Personal Assistant and several other agents have the capability to spawn agents to complete tasking requested by the user. For example, the user requests that an Internet search be performed to gather information on software design tools that are not currently in the OPAD. The assistant would task the Interactive Tutorial Coordinator with this task. The Interactive Tutorial Coordinator would first query the Tools Agent for the current tool set. It would then spawn a mobile agent with basic query information, taking into account the tool exception list received from the Tools Agent to accomplish an Internet search via the Access Internet Facilitator. Once the mobile agent returns with the information, it is forwarded to the User Personal Assistant for display. The user would see a ranked list of search hits matching the original query.

MENTOR also has the ability to learn from its user through user interface interactions, user feedback, and agent to agent interactions. This will allow the system to help the users build more efficient processes and optimize itself for future use.

E. CONCEPTUAL MENTOR SYSTEM ARCHITECTURE

MENTOR will run on a system that is client-server configured. It consists of one personal computer per user, the client, with shared data being kept on one or more file servers. The conceptual MENTOR System Architecture is shown in Figure 5.4.

Each client has their own User Personal Assistant, with a supporting Setup User Interface agent that allows the user to update their interface and Specific Application agents that interface with local applications residing on the client (i.e. MS Office Suite,

MS Project, and Outlook). The MENTOR System Server houses the main team of MENTOR agents. Furthermore, the OPAD and individual PPADs each reside on their own server. Agents residing on the individual clients and servers communicate using either Remote Procedure Calls or Remote Programming.

Communications between stationary agents is done through Remote Procedure Calls (RPC) and is the same procedure as calling a remote software module from a main software program. This client-server type communication principle sends request messages for specific procedures to be performed. Once the remote procedures are performed, the results are transferred back to the requesting agent in a reply message. In contrast, mobile agents use Remote Programming (RP) principles to communicate with other objects in the system. RP transfers the client's calling procedure to the server where is executed locally. This contributes to the advantage that mobile agents have for minimizing network load. This provides obvious payoffs when there are multiple simultaneous users. Generally, there will be multiple clients, and therefore, the system configuration will look like the one shown in Figure 5.5.

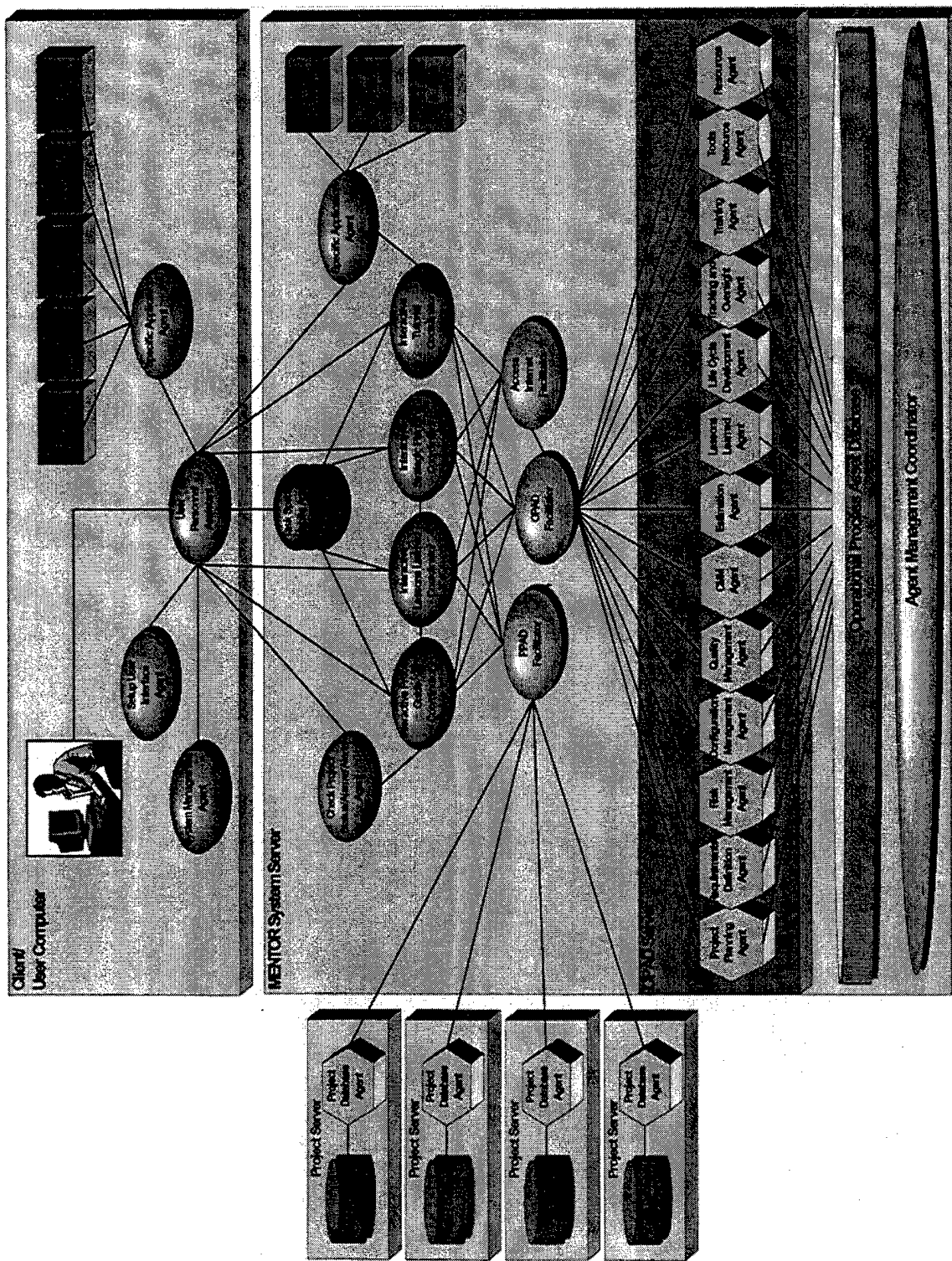


Figure 5.4. Conceptual MENTOR System Architecture

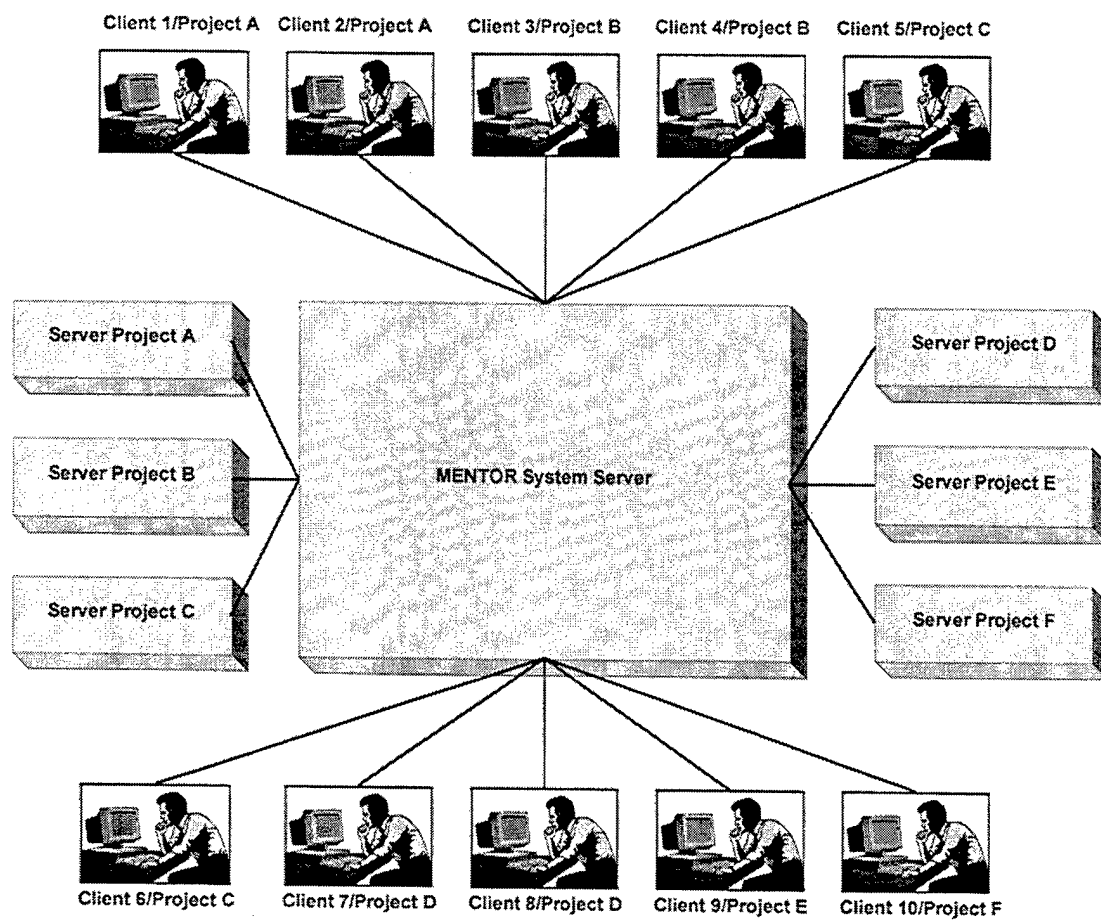


Figure 5.5. MENTOR Client/Server Configuration

VI. SOFTWARE AGENT CASE SCENARIO

MENTOR can benefit the software development team in managing software development tasks in many ways. Performing cost, size, and schedule estimations are just one example where MENTOR will contribute. Sound estimations are essential to the successful completion of a software development project. MENTOR will be able to speed up the estimation process by helping the development team gather estimation data and lessons learned from other similar projects, then use this information to help the team develop sound realistic estimations. It will also provide an interface to the estimation tools that are standard to the SPAWAR organization. This chapter shows how MENTOR can help the development team perform estimations by using a UML role-playing scenario to show MENTOR agent interactions.

Before we can investigate how MENTOR will help the team perform cost, schedule, and size estimations, a baseline for the manual estimating process must be outlined. Based on this outline, an estimation role-playing scenario will be developed. The baseline and role playing scenarios are based on a hypothetical software package that is outlined in Section A of this chapter. In addition, a brief background on the hypothetical software development team performing the estimation is given in Section B.

A. TEAM BACKGROUND AND ASSUMPTIONS

The hypothetical team that will be performing the estimations and using MENTOR is representative of many of the development teams at SPAWAR. The development team has members who have written code on previous projects and/or developed programs on their own or on a team. Some are even new engineers just

starting out. Most of the team is inexperienced in the estimation process. Some do not even realize that there is a SPAWAR approved estimation process. The team members that have performed some form of cost, size, and schedule estimation have not had positive experiences.

Previous project estimations all seemed to leave them short on money and schedule and often resulted in having to go back to the sponsor for additional funds and an increase in the performance period. The estimation process they followed consisted of one person sitting down and guessing at costs and schedule based on their own past experiences, sometimes not even considering the size of the software, the functions the software should provide, or the other team member's experiences and inputs. This led to an estimate that often optimistic, but unrealistic. In order to overcome some of these estimating problems, several of the development team members attended the Software Program Managers Course given by the SEPO.

During this course, they learned valuable information that could be used to manage a software project. They also learned of the SEPO web site that contains all the SPAWAR policies and processes for each phase of the software life cycle. It is assumed that they will use this web site as a resource in the estimation of the new software project outlined in the next section.

B. HYPOTHETICAL SOFTWARE PACKAGE

The information given in this section is the typical information that is provided to a software development team when asked to provide estimates to a sponsor on a new

project. A hypothetical software problem statement used as the basis for this chapter's scenarios is outline below:

A software development team is tasked to develop a software package that provides a graphical user interface for inputting government Credit Card Purchases (CCPs) and automatically routing purchases through the system. A review of the system specification reveals that the software will run on a PC connected to a LAN and must interface with peripherals such as a monitor, mouse, scanner, and laser printer. The top-level scope of the effort is as follows:

The software will accept purchase card data from the user. The user will interact and control the CCP system through a user interface that has human interface considerations. All CCP data and supporting information will be maintained in a CCP database. Modules will be developed that route the CCPs through the purchase process and output reports based on individual credit card holders or groups of credit card holders. The software will be designed to interface and control devices such as a monitor, mouse, scanner, and laser printer. The software package will be developed using object-oriented methodologies and programming languages.

This information will be used as the basis for the scenarios in the sections that follow.

C. MANUAL ESTIMATION BASELINE SCENARIO

This estimation baseline begins with the team meeting to discuss the new tasking. They decide that they are going to start using the SEPO estimation process that they learned about in the SEPO Software Program Managers Course. Unfortunately, no one can remember exactly how to proceed. So, they decide to search the SEPO Web site for information. They find the SEPO's Software Estimation Process. This process suggests using two methods of estimation for comparison and verification purposes. If the values are within 20% of each other, they are acceptable; otherwise an investigation is performed for reasons of discrepancy. In addition, the Software Estimation Process

outlines methods for developing estimates for size, effort, cost, and schedule. The first step in the estimation process is to choose two methods of estimation.

They chose to use Lines of Code (LOC) based estimation and an automated tool called REVIC, which is available through the SEPO Web site. After reviewing the LOC method, they began by deciding what major functional areas would be needed for the new project. They chose the six major functional areas listed below based on the initial project description, as outlined above:

- User Interface
- Database Management
- Graphics and Display
- Peripheral Interface
- Routing Module
- Reports Module

Based on these areas, the team developed low, nominal and high LOC values for each of the functional areas they identified. Because only a couple of development team software engineers had experience in previous software development efforts, the team decided to do some research within the organization for projects with similar requirements, in hopes of developing more reliable estimates for LOC. Unfortunately, the team found this was not as easy as it seemed.

The organization as a whole was so diverse, that finding other similar projects with similar functional areas was challenging. They began by making phone calls to other software projects, asking the Lead Software Engineers for LOC estimates on functional areas that were similar to their own. Most of the replies yielded the same answer, metrics of this kind were not kept, but source code could be provided for the team to count and gather the information for themselves. After receiving the source

documentation, the team began counting LOC. This tedious task of contacting other projects, waiting to receive the source documentation, then counting LOC took approximately 2 weeks. Finally, with low, nominal, and high values for LOC in hand, the team applied these values to the LOC-based estimation method. Then, they established an estimate for total LOC. Given the past experiences of team members, and taking into account the relative inexperience of the team as a whole, they arrived at a productivity of 700 LOC/pm. Next, using the organizations burdened labor rate per month, they found the cost per line of code. Based on these values, they arrived at an estimated cost and effort in person months. However, they had only completed one of the two estimation methods.

The second method involved using an automated estimation tool called REVIC, which is based on the Revised Intermediate COCOMO model. Because only a few of the team members had used the tool briefly in the Software Program Managers course, they had to learn how to use the tool all over again. Once they were familiar with REVIC they began the second estimation for the new project.

The first step was to specify values ranging from very low to very high for the 20 environmental factors REVIC indicated. These included factors such as programmer capability, applications experience, modern programming practices, and program language experience. REVIC then asked for the module names, representing the functional areas the team had chosen. Next, they specified low, most probable, and high estimates of Delivered Source Instructions, using the same LOC research values that were used in the manual LOC-based method. REVIC then provided an estimate for

effort, cost, and schedule. They then compared the two estimates and found the values to be within 20% of each other.

D. MENTOR ESTIMATION SCENARIO

The role-playing scenario used in this section follows the baseline scenario in section C above. It shows the User-to-UPA interactions, the associated actions and interactions between MENTOR agents, and the source and target objects for those interactions. The role-playing scenario models "order dependent message sequences among objects collaborating to produce system behavior." [Ref. 6] This tool also provides a means to validate the MENTOR concept and its expectations. The MENTOR estimation role-playing scenario follows in Table 6.1, with the role-playing scenario legend shown in Table 6.2.

MENTOR Estimation Role-Playing Scenario				
Step	User-to-User Assistant Interactions	MENTOR Action	Source	Target
	[Initial System State – UPA has been setup and initial new project data has been entered. UPA waiting for user input]			
1	User requests assistance on estimating size, cost, effort and schedule for a new project.	UPA processes query.	User	UPA
2	UPA asks the user if they would like to review the tutorial on estimation, review the lessons learned on estimation, or perform project estimation.	UPA responds to user.	UPA	User
3	User selects perform project estimation.	User responds to UPA.	User	UPA
4	One moment while retrieving estimation information.	UPA responds to user.	UPA	User
5		UPA processes query then queries IPGC for assistance with the estimation process.	UPA	IPGC
6		IPGC replies with an affirmative message.	IPGC	UPA

Table 6.1. MENTOR Estimation Role-Playing Scenario

MENTOR Estimation Role-Playing Scenario				
Step	User-to-User Assistant Interactions	MENTOR Action	Source	Target
7		UPA Tasks IPGC with estimation task.	UPA	IPGC
8		IPGC queries OF for agents that possess the information needed to complete the estimation task.	IPGC	OF
9		OF replies with passes to the EA, LLA, and RA agents.	OF	IPGC
10		IPGC travels to EA.	IPGC	EA
11		IPGC queries EA for estimation process data, estimation algorithms, references to automated estimation tool agents, and initial criteria list the user must provide.	IPGC	EA
12		EA responds with data.	EA	IPGC
13		IPGC requests UPA to ask user which two estimation methods, from the provided list, the user would like to use.	IPGC	UPA
14	Select two estimation methods for your new project.	UPA displays list of estimation methods and asks user to select two methods.	UPA	User
15	User selects LOC-based estimation and the REVIC automated tool method.	User selects two methods.	User	UPA
16		UPA replies to IPGC with user information of LOC-based estimation and REVIC.	UPA	IPGC
17		IPGC reviews the estimation process and methods.	IPGC	
18		IPGC requests the UPA to ask the user for a list of the major functional areas for the new package.	IPGC	UPA
19	List the major functional areas of your new software package.	UPA displays request.	UPA	User
20	User responds with a list: User Interface, Database Management, Graphics and Display, Peripheral Interface, Routing Module, and Reports Module.	User enters a list of functional areas.	UPA	IPGC
21		UPA replies to IPGC with user's major functional areas.	UPA	IPGC
22		IPGC queries EA for estimation data regarding LOC on these types of functional areas.	IPGC	EA
23		EA replies with data.	EA	IPGC

Table 6.1 Continued. MENTOR Estimation Role-Playing Scenario

MENTOR Estimation Role-Playing Scenario				
Step	User-to-User Assistant Interactions	MENTOR Actions	Source	Target
24		IPGC reviews information for applicability based on current task requirements and discards unneeded information.	IPGC	
25		IPGC travels to LLA.	IPGC	LLA
26		IPGC queries LLA for estimation data regarding projects with similar functional areas.	IPGC	LLA
27		LLA replies with data.	LLA	IPGC
28		IPGC reviews information for applicability based on current task requirements and discards unneeded information.	IPGC	
29		IPGC travels to RA.	IPGC	RA
30		IPGC queries RA for estimation data on past projects within the organization.	IPGC	RA
31		RA replies with data.	RA	IPGC
32		IPGC reviews information for applicability based on current task requirements and discards unneeded information.	IPGC	
33		IPGC reviews all information regarding LOC and forms estimates for LOW, NOMINAL, and HIGH LOC.	IPGC	
34		Using the burdened labor rate and the organizational productivity average for LOC/pm obtained through research and the LOC estimates for LOW, NOMINAL, and HIGH the IPGC then performs calculation for estimated total LOC, effort, cost, and schedule.	IPGC	
35		IPGC requests the UPA to display the estimation data from the LOC-based estimation method.	IPGC	UPA
36	UPA displays estimation data for LOW, NOMINAL, and HIGH LOC for all functional areas, burdened labor rate, productivity average, total estimated LOC, effort, cost, and schedule.	UPA displays information.	UPA	User
37		IPGC requests UPA to tell user to stand by for Interactive REVIC estimation.	IPGC	UPA
38	Please stand by for interactive REVIC estimation.	UPA displays request.	UPA	User

Table 6.1 Continued. MENTOR Estimation Role-Playing Scenario

MENTOR Estimation Role-Playing Scenario				
Step	User-to-User Assistant Interactions	MENTOR Actions	Source	Target
39		IPGC requests assistance from a REVIC SAA and stands by for assistance to SAA.	IPGC	REVIC SAA
40		SAA initiates REVIC Interface.	SAA	
41		REVIC SAA requests UPA to have user select values from the range VL, LO, NM, HI, and VH for each environmental factor in the list.	REVIC SAA	UPA
42	UPA displays the environmental factor list and requests that the user selects a value, from the range pull down menu, for each of the environmental factors listed.	UPA displays information and requests range information.	UPA	User
43	User selects range information using pull-down menus for each environmental factor listed.	UPA replies to REVIC SAA with information.	UPA	REVIC SAA
44		REVIC SAA requests major functional area data with LOW, NOMINAL (Most Probable), and HIGH LOC data and burdened labor rate from IPGC.	REVIC SAA	IPGC
45		REVIC SAA enters data into REVIC tool for estimate calculation.	REVIC SAA	REVIC
46		REVIC SAA extracts calculation data from REVIC tool.	REVIC SAA	REVIC
47		REVIC SAA provides estimation data on environmental factors, range values selected, major functional areas, LOC LOW, NOMINAL, and HIGH values, total LOC, effort, cost, and schedule to UPA for display and to IPGC for reference.	REVIC SAA	UPA/ IPGC
48		SAA has completed task and shuts down.	SAA	
49	UPA displays the REVIC estimation data on environmental factors, range values selected, major functional areas, LOC LOW, NOMINAL, and HIGH values, total LOC, effort, cost, and schedule.	UPA displays information to user.	UPA	User
50		IPGC requests UPA to stand by for comparison report.	IPGC	UPA
51	Stand by for comparison report.	UPA displays information to user.	UPA	User

Table 6.1 Continued. MENTOR Estimation Role-Playing Scenario

MENTOR Estimation Role-Playing Scenario				
Step	User-to-User Assistant Interactions	MENTOR Actions	Source	Target
52		IPGC performs comparison calculations, determines the estimates to be within 20% of each other, and formulates a report with both estimation method data.	IPGC	
53		IPGC requests the UPA to display estimation comparison report to user and acceptable within 20% of comparison results.	IPGC	UPA
54	UPA displays completed estimation report. Estimates are within 20% of each other and recommended as acceptable.	UPA displays information to user.	UPA	User
55		IPGC requests UPA to ask user if this estimation is to be stored in the new project database.	IPGC	UPA
56	Would you like to store this estimate in the new project database?	UPA displays information to user.	UPA	User
57	Yes.	User responds to UPA.	User	UPA
58		UPA replies to IPGC with user answer.	UPA	IPGC
59		IPGC sends request to PF to store the attached file in the database under estimation data.	IPGC	PF
60		PF passes information to PDA requesting storage of attached file under estimation data.	PF	PDA
61		PDA processes request, stores file, sends affirmative reply to PF.	PDA	PF
62		PF responds affirmatively to IPGC.	PF	IPGC
63		IPGC requests UPA to displays information regarding the storage of estimation report.	IPGC	UPA
64	File has been stored under estimation data in the new project database.	UPA displays information.	UPA	User
65		IPGC incorporates any new information into its knowledge database, sends a task complete message to UPA then shuts down.	IPGC	
	[UPA waiting for user input]			

Table 6.1 Continued. MENTOR Estimation Role-Playing Scenario

MENTOR Estimation Role-Playing Scenario Legend	
Acronym	Description
EA	Estimation Agent
IPGC	Interactive Process Guide Coordinator
LLA	Lessons Learned Agent
OF	OPAD (Operational Process Asset Database) Facilitator
PF	PPAD (Project Process Asset Database) Facilitator
RA	Resource Agent
REVIC SAA	REVIC Specific Applications Agent
SAA	Specific Applications Agent
UPA	User Personal Assistant

Table 6.2. Legend for MENTOR Estimation Role-Playing Scenario

This scenario shows the feasibility of the MENTOR agent team to perform tasks, such as software estimation. And, even though the scenario above appears to be quite extensive, the time that MENTOR will take to complete the same tasking that was performed manually in Section C above, is considerably less, with an estimate for this task being less than one day. In addition, because the MENTOR system agents are autonomous, they do not require constant monitoring by the user. Instead, the user tasks the UPA agent with the estimation task, then the user proceeds with their own daily routine. MENTOR will notify the user when the tasking is completed. Furthermore, the MENTOR scenario did not require the user to be an expert at the estimation task. Rather, it asked the user strategic questions then completed the tedious task of gathering data on the estimation process, LOC estimates for similar projects, and calculating the actual estimate on its own. This is not to say that MENTOR should replace the knowledge a team should have to estimate a software project, but that it should alleviate mundane tasks such as gathering information, and ensure that all factors are taken into account. This is evident in the MENTOR scenario when the IPGC searched several of the OPADs to find similar project LOC data and lessons learned. However, MENTOR can facilitate

this process and guide the user in the right direction, but the user is ultimately responsible for the estimation and should therefore review the estimate, as well as seek other team members inputs of the estimate.

VII. SUMMARY AND RECOMMENDATIONS

This thesis provides a conceptual agent architecture design for an intelligent agent network that provides decision support for teams managing software development efforts. The MENTOR System was proposed to aid the software development manager and team in successfully learning, planning, managing, and troubleshooting the software development process and its tasks. MENTOR accomplishes this by using a network of autonomous intelligent agents that collaborate to provide the user with the knowledge base and assistance that is integral to providing a repeatable, defined, managed, and optimized development environment to a diversified organization like SPAWAR Systems Center, San Diego.

The conceptual MENTOR network structure presented in this thesis is composed of individual user interfaces and a network of decentralized supporting agents that facilitate software development task completion in an on-demand basis. Software tasking includes process guidance, lessons learned consultation, strategic planning projections, process tutelage, application intercommunications, and team communications. The knowledge base which the MENTOR agent network uses to gather information and base decisions on is vast, covering software development areas for all phases of the development process. Since this is a dynamic learning environment, additional knowledge bases can be added at any time to update the user-agent knowledge base resulting in instantaneous organizational information flow. This means that the right people get the right information at the right time in order to achieve positive results in their software development efforts.

This type of dynamic software development environment is recommended and will eventually be essential to a successful software development house, such as SPAWAR Systems Center. Future work for the continuation of the MENTOR effort is outlined in the following chapter of this thesis.

VIII. FUTURE WORK

A. DETAILED ANALYSIS AND DESIGN

A step to further MENTOR development would be to perform a detailed requirements analysis for the MENTOR system. Since this thesis only addresses a conceptual model and its requirements, a detailed requirements analysis remains to be performed. The objects that comprise the system should be specified in terms of attributes and behaviors in order to define the structure and dynamics of the system. [Ref. 6] In addition, class diagrams should be developed, as well as class relationships, associations, data structures and algorithms for each class. Further analysis should include modeling the system with state diagrams, including detailed interaction scenarios that identify the services, communications, and control relationships between the agents. A prototype should be developed early in the detailed design process to help refine the object classes, data structures, and algorithms needed to complete a working model.

B. FIRST PHASE SYSTEM IMPLEMENTATION

An obvious first step in MENTOR system development is to implement a working prototype for the Interactive Tutorial portion of the MENTOR system. This will involve the User Personal Assistant, Interactive Tutorial Coordinator, OPAD Facilitator, and all OPAD supporting agents. Because of the basic search and retrieve functions inherent in the interactive tutorial and the availability of intelligent software information agents, this MENTOR functionality would be the easiest to implement in the shortest period of time. However, the knowledge base, which MENTOR utilizes will require

detailed investigation as to the exact contents, methods for storage and retrieval, and format requirements for stored information to allow maximum information storage and retrieval efficiency. This module will also provide immediate information flow by allowing team members, who do not have current knowledge or the time to attend classes, to access current up-to-date knowledge at anytime, based on a mere request for help from MENTOR.

C. SYSTEM SECURITY

Security considerations for agent-based technologies are double-edged swords. On one hand, the agents must be free to complete their tasking with access to remote machines and databases. On the other hand, the agent must be controlled so that it does not cause more harm than good. It is also important that the system be protected from outside attacks, as well as a means to identify whether an agent or user is friend or foe. Therefore, it will be important to the success of MENTOR that security policy be articulated and enforced to ensure that data is protected from being modified, tampered with, or deleted by unauthorized users, and that agent-to-agent communications are trusted.

D. INTEGRATION OF OTHER SPAWAR THESIS EFFORTS INTO THE OPAD

Ongoing thesis efforts at SPAWAR in the areas of quality management metrics [Ref. 26] and the software evolution process for COTS components used in military applications [Ref. 27] should be investigated for further incorporation into the MENTOR system-agent architecture. Research efforts in both areas would provide valuable knowledge that

MENTOR could utilize in continually improving its mentoring of software development teams.

E. HIGH PERFORMANCE ORGANIZATION MODEL

SPAWAR Systems Center recently adopted a new plan for implementing high-performance principles as a means of improving the organization. Future work could assess how MENTOR would implement High Performance Organization (HPO) principles and guide the user through the underlying process. MENTOR could provide guidance on developing a business unit's Mission and Leadership Functions, such as Vision and Values and incorporate methods for strategic business planning using the HPO Model. Leadership philosophy assessments of individual business units could also be determined based on real-time team evaluations that are maintained by MENTOR. Once the leadership philosophy is assessed, MENTOR can guide the unit to the leadership level desired. HPO principles also suggest an organization that is based on a Network Talent Model. This organizational model is not based on the typical hierarchical organizational structure, but instead is a network of Naturally Occurring Groups (NOGs) that form business units. [Ref. 13] As SPAWAR converts from a Hierarchical type organization to the Network Talent Model type organization, a parallel organization structure must exist between both. MENTOR could maintain the structural connections between the two and facilitate a more seamless transition. An HPO database and artifacts repository must also be developed for incorporation into the OPAD. This database will provide resources, samples, contacts, lessons learned, and guidance on all HPO principles and strategies.

F. SEI CERTIFICATION

Another important initiative at SPAWAR is for the organization to attain certification from the Software Engineering Institute (SEI). This process certifies an organization at one of the five CMM levels. SPAWAR's goal is to become CMM Level 3 certified. MENTOR has the ability to guide the software projects through this process by making sure all the criteria are met for certification. Future work would consist of developing the SEI certification criteria for the organization and incorporating the knowledge and rule-base into the OPAD and an Interactive SEI Facilitator Agent into the agent architecture.

LIST OF REFERENCES

1. ACM SIGSOFT, *Software Engineering Notes*, Vol. 15, No. 5, October 1990, pp. 50-59.
2. Bradshaw, Jeffrey M. ed, *Software Agents*, Menlo Park, California: American Association for Artificial Intelligence, 1997.
3. Brenner, Walter et al., *Intelligent Software Agents: Foundations and Application*, Germany: Springer-Verlag, 1998.
4. Brooks, Frederick P., *The Mythical Man-Month*, Reading, Massachusetts: Addison-Wesley, 1995.
5. Bui, Tung, "Building agent-based corporate information systems: an application to telemedicine," *European Journal of Operational Research*, Article No. 4011: September 1999, pp 1-16.
6. Douglass, Bruce Powel, *Real-Time UML*. Reading, Massachusetts: Addison Wesley Longman, Inc., 1998.
7. Gates, Bill, *Business@The Speed of Thought*, New York, New York: Warner Books, Inc., 1999.
8. Huhns, Michael N. and Munindar P. Singh, ed., *Readings in Agents*. San Francisco, California: Morgan Kaufmann Publishers, Inc., 1998.
9. Osmundson, John, *IS4300 Software Project Managent Course Class Notes*, Naval Postgraduate School, 1998.
10. Paulk, M. et al., *Capability Maturity Model for Software*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1993.
11. Paulk, M. et al., *The Capability Maturity Model for Software, Version 2B*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1997.
12. Phillips, Dwayne. *The Software Project Manager's Handbook, Principles That Work At Work*, Los Alamitos, California: IEEE Computer Society Press, 1998.

13. Pickering, J., *Building High-Performance Organizations for the Twenty-First Century*, Commonwealth Center for High-Performance Organizations, Inc. and The Federal Executive Institute, 1999.
14. Pressman, Roger S., *Software Engineering. A Practitioner's Approach*. Fourth Edition. New York, NY: McGraw-Hill, 1997.
15. Software Engineering Process Office, *Requirments Management Guidebook*, SPAWAR Systems Center, San Diego, California, 1998.
16. Software Engineering Process Office, *Software Management for Executives Guidebook*, SPAWAR Systems Center, San Diego, California, 1999.
17. Software Engineering Process Office, *Software Project Management Course Notes*, SPAWAR Systems Center, San Diego, California, 1999.
18. Software Engineering Process Office, *Software Project Planning Process*, SPAWAR Systems Center, San Diego, California, 1999.
19. Software Engineering Process Office, *Risk Management Process*, NCCOSC RDT&E Division, 1997.
20. Software Engineering Process Office, *SEPO Homepage*, <http://sepo.spawar.navy.mil>, SPAWAR Systems Center, 2000.
21. Software Engineering Process Office, *Software Estimation Process*, SPAWAR Systems Center, San Diego, California, 1999.
22. Software Engineering Process Office, *Software Project Tracking and Oversight Process*, SPAWAR Systems Center, San Diego, California, 1999.
23. SPAWAR Systems Center, *Information and Decision Management: Tools for Decision-Makers*, SPAWAR Systems Center, San Diego, California, July 1998.
24. Tecuci, Gheorghe, *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*, San Diego, California: Academic Press, 1998.
25. Weld, D., "The Role of Intelligent Systems in the National Information Infrastructure," *AI Magazine*, Fall 1995, pp. 45-64.
26. Machniak, Martin J., *Development Of A Quality Management Metric (QMM) Measuring Software Program Management Quality*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1999.

27. Hensley, Barry J., *Development Of A Software Evolution Process For Military Systems Composed Of Integrated Commercial Off The Shelf (COTS) Components*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 2000.

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

1. Ames et al., "Applications of Web-Based Workflow," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 1: 1998.
2. Arcelli, F. et al., "Software Agents for Computer Vision: A Preliminary Discussion," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 5: 1998.
3. Atkins, Daniel E. et al., "Toward Inquiry-Based Education Through Interacting Software Agents," *IEEE Computer*, Vol. 29, No. 5: May 1996, pp 69-76.
4. Bellika, Johan Gustav et al., "Using User Models in Software Agents: The Virtual Secretary," *Proceedings of the 3rd International Conference on Multi-Agent Systems*, 1998.
5. Berger, Michael et al., "A Learning Component for Workflow Management Systems," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 7: 1998.
6. Buhr, R. J. A. et al., "A High Level Visual Notation for Understanding and Designing Collaborative, Adaptive Behavior in Multiagent Systems," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 6: 1998.
7. Bui, Tung and Siva Sankaran, "Group Decision and Negotiation in Telemedicine: An Application of Intelligent Mobile Agents as Non-Human Teleworkers," *Proceedings of the 30th International Conference on Systems Sciences*, Maui, Hawaii, January 1997.
8. Burstein, Mark. H. et al., "An Approach to Mixed-Initiative Management of Heterogeneous Software Agent Teams," *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.
9. Chen, Hsinchun et al., "Toward Intelligent Meeting Agents," *IEEE Computer*, Vol. 29, No. 8: August 1996, pp 62-70.
10. Clement, Bradley J. and Edmund H. Durfee, "Scheduling High-Level Tasks Among Cooperative Agents," *Proceedings of the 3rd International Conference on Multi-Agent Systems*, 1998.

11. Desbiens, Jocelyn et al., "Communication and Tracking Infrastructure of a Mobile Agent System," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 7: 1998.
12. Etzioni, Oren and Daniel S. Weld, "Intelligent Agents on the Internet: Fact, Fiction, and Forecast," *IEEE Expert/Intelligent Systems & Their Applications*, Vol. 10, No. 4: August 1995, pp. 44-49.
13. Huhns, Michael N. and Munindar P. Singh, "Agents On The Web," *IEEE Internet Computing*, Vol. 1, No. 3: May/June 1997, pp. 80-82.
14. Humphrey, Watts S., "Three Dimensions of Process Improvement, Part I: Process Maturity," *Crosstalk*, STSC, Hill Air Force Base, UT 84056-5205, February 1998.
15. Humphrey, Watts S., "Three Dimensions of Process Improvement, Part II: The Personal Process," *Crosstalk*, STSC, Hill Air Force Base, UT 84056-5205, March 1998.
16. Humphrey, Watts S., "Three Dimensions of Process Improvement, Part III: The Team Process," *Crosstalk*, STSC, Hill Air Force Base, UT 84056-5205, April 1998.
17. Jehuda, Jair, "Collaborative Best-Effort Real Time Agents - A New Paradigm," *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.
18. Joshi, Niraj and V.C. Ramesh, "Intelligent Agents for Internet-Based Military Training," *Crosstalk*, STSC, Hill Air Force Base, UT 84056-5205, Mar 1998.
19. Knapik, Michael and Jay Johnson, *Developing Intelligent Agents for Distributed Systems: Exploring Architecture, Technologies, and Applications*, New York, NY: McGraw-Hill, 1998.
20. Knotts, Gary et al., "A Project Management Tool for Computer-Supported Cooperative Work During Project Planning," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 1: 1998.
21. Krulwich, Bruce, "AUTOMATING THE INTERNET: Agents as User Surrogates," *IEEE Internet Computing*, Vol. 1, No. 4: July/August 1997, pp 34-38.

22. Kulpa, Margaret, "Ten Things Your Mother Never Told You About the Capability Maturity Model," *Crosstalk*, STSC, Hill Air Force Base, UT 84056-5205, September 1998.
23. Labrou, Yannis et al., "The Interoperability Problem: Bringing Together Mobile Agents and Agent Communication Languages," *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.
24. Lai, Hsiangchu and Tzyy-Ching Yang, "A System Architecture of Intelligent-Guided Browsing on the Web," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 4: 1998.
25. Macintosh, Ann, "A Profile of AIAI," *IEEE Expert*, Vol. 10, No. 3: June 1995, pp. 4-5, 78-80.
26. Maes, Pattie, "PATTIE MAES ON SOFTWARE AGENTS: Humanizing the Global Computer," *IEEE Internet Computing*, Vol. 1, No. 4: July/August 1997, pp10-19.
27. Mahling, Dirk E. et al., "From Office Automation to Intelligent Workflow Systems," *IEEE Expert*, Vol. 10 No. 3: June 1995, pp. 41-47.
28. Milojicic, Dejan S. et al., "Agents: Mobility and Communication," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 7: 1998.
29. Mizoguchi, Riichiro and Hiroshi Motoda, "Expert Systems Research in Japan," *IEEE Expert*, Vol. 10, No. 4: August 1995, pp 14-23.
30. Murch, Richard and Tony Johnson, *Intelligent Software Agents*, Upper Saddle River, NJ: Prentice Hall PTR, 1999.
31. Nangsue, Phadern and Susan E. Conry, "Fine-Grained Multiagent Systems for the Internet," *Proceedings of the 3rd International Conference on Multi-Agent Systems*, 1998.
32. O'Leary, Daniel E., "Using AI in Knowledge Management: Knowledge Bases and Ontologies," *IEEE Intelligent Systems & Their Applications*, Vol. 13, No. 3: May/June 1998, pp 34-39.
33. Rumbaugh, James et al., *The Unified Modeling Language Reference Manual*, Reading Massachusetts: Addison Wesley Longman, Inc., 1999.

34. Sengupta, Kishore and J. Lean Zhao, "Designing Workflow Management Systems for Virtual Organizations: An Empirically Grounded Approach," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 4: 1998.
35. Sheppard, John W. and Gerald C. Hadfield, "The Object-Oriented Design of Intelligent Test Systems," *Crosstalk*, STSC, Hill Air Force Base, UT 84056-5205, August 1994.
36. Singh, Munindar P., Michael N. Huhns, "Internet-Based Agents: Applications and Infrastructure," *IEEE Internet Computing*, Vol. 1, No. 4: July/August 1997, pp. 8-9.
37. Singh, Munindar, "Developing Formal Specifications to Coordinate Heterogeneous Autonomous Agents," *Proceedings of the 3rd International Conference on Multi-Agent Systems*, 1998.
38. Sorensen, Reed, "Can Your Process Benefit From Workflow Products?," *Crosstalk*, STSC, Hill Air Force Base, UT 84056-5205, February 1996.
39. Stillman, Johnathan and Piero Bonissone, "Developing New Technologies for the ARPA-Rome Planning Initiative," *IEEE Expert*, Vol. 10, No. 1: February 1995, pp. 10-16.
40. Suguri, Hiroki, "A Standardization Effort for Agent Technologies: The Foundation for Intelligent Physical Agents and Its Activities," *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.
41. Sycara, Katia et al., "Distributed Intelligent Agents," *IEEE Expert/Intelligent Systems & Their Applications*, Vol. 11, No. 6: December 1996, pp 36-46.
42. Textel, Putnam P. and Charles B. Williams, *Use Cases Combined With Booch OMT UML: Process and Products*, Upper Saddle River, NJ: Prentice Hall PTR, 1997.
43. Tu, Hsieh-Chang and Jieh Hsiang, "An Architecture and Category Knowledge for Intelligent Information Retrieval Agents," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 4: 1998.
44. Wang, John Chia-chin, "A Framework for Resolving Multiagent Collaboration Dilemmas," *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 4: 1998.

45. Yeung, Chris and Pang-Fei Tung, "A Multi-agent Based Tourism Kiosk on Internet," *Preceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 4: 1998.
46. Zarate, P. and C. Rosenthal-Sabroux, "A Cooperative Approach for Intelligent Decision Support Systems," *Preceedings of the 31st Annual Hawaii International Conference on System Sciences*, Vol. 5: 1998.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center2
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library2
Naval Postgraduate School
411 Dyer Road
Monterey, California 93943-5101

3. Chairman, Code CS1
Naval Postgraduate School
Monterey, California 93943-5100

4. Dr. Luqi, CS/Lq1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5100

5. Dr. J. Bret Michael, Code CS/Mj1
Computer Science Department
Naval Post Graduate School
Monterey, California 93943-5100

6. Dr. John Osmundson, Code CC/Os1
C3 Academic Group
Naval Postgraduate School
Monterey, California 93943-5100

7. Software Engineering Process Office, Code D121
SPAWAR Systems Center
53560 Hull Street
San Diego, California 92152

8. Navigation and Applied Sciences Department, Code D301
SPAWAR Systems Center
53560 Hull Street
San Diego, California 92152

9. Lori A. Church2
10797 Carillon Court
San Diego, California 92131